MR MAQ : ALGORITME DE READ MAPPING UTILITZANT LA PLATAFORMA HADOOP

Memòria del Projecte Fi de Carrera
d'Enginyeria Superior en Informàtica
realitzat per
Joan Protasio Ramirez
i dirigit per
Porfidio Hernández Budé
Bellaterra, Juny de 2010

Universitat Autònoma de Barcelona

etse Escola Tècnica Superior d'Enginyeria

El sotasignat, Porfidio Hernández Budé

Professor de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Joan Protasio Ramirez.

I per tal que consti firma la present.

Signat: Porfidio Hernández Budé

Bellaterra, Juny de 2010

"Y cuando Alejandro vió la inmensidad de sus dominios lloró porque no había más mundos por conquistar"

John Milton sobre un passatge de Alejandro Magno

Agraïments

La carrera d'un estudiant d'Enginyeria Superior Informàtica és molt llarga i laboriosa. És un bonic i llarg camí que requereix de molta dedicació i d'una motivació continua. És per això que amb aquestes paraules voldria agrair a tothom l'ajuda i el suport que m'han donat. Sobretot als meus pares, els que han fet possible que a dia d'avui hagi arribat on estic. A la meva germana Mª José per fer-me riure en els dies durs d'exàmens i al meu germà Jerva per donar-me exemple amb la seva carrera també com Enginyer Electrònic. També a en Sergi i la Mònica i en especial a la meva novia Gladis, per animar-me dia a dia i entendre'm ens els dies llargs de tant treball amb les pràctiques.

També voldria donar les gràcies als professors, i ara companys, Juan Carlos Moure i Toni Espinosa els quals m'han ajudat a millorar el PFC des de moltes i diferents perspectives. I finalment al meu director en Porfidio Hernández, per la seva gran ajuda i pels seus grans coneixements que m'han orientat molt positivament al llarg de tot el Projecte.

Signat : Joan Protasio Ramirez

Bellaterra, Juny de 2010

Índex

Capítol I

Introducció

La **Bioinformàtica** va néixer de la necessitat de la **Biologia** per resoldre problemes d'una complexitat temporal i espacial elevada en càlculs matemàtics. Aquest tipus de problemàtica, com podrem veure en el següent capítol amb més detall, té una naturalesa altament compatible amb les solucions que aporta l' Enginyeria Informàtica.

Per aquest motiu l'any 1979 *Paulien Hogeweq* va batejar la *Bioinformàtica* com la disciplina científica que aplica el potencial de l'enginyeria informàtica per resoldre problemes d'origen biològic.

Com totes les especialitats, la *Bioinformàtica* disposa de nombroses branques, però la podem simplificar com la ciència que resol problemes de caire **genètic** i **genòmic**. Aquesta última branca, la **Genòmica** es defineix com la ciència que estudia el funcionament, l' evolució i l' origen dels **genomes.**

El *Genoma*, en el nostre cas el **Genoma humà,** és anomenat més popularment com l'**ADN** o **DNA** i es tracta del "recull" de tota la **informació genètica** que disposa un ésser viu.

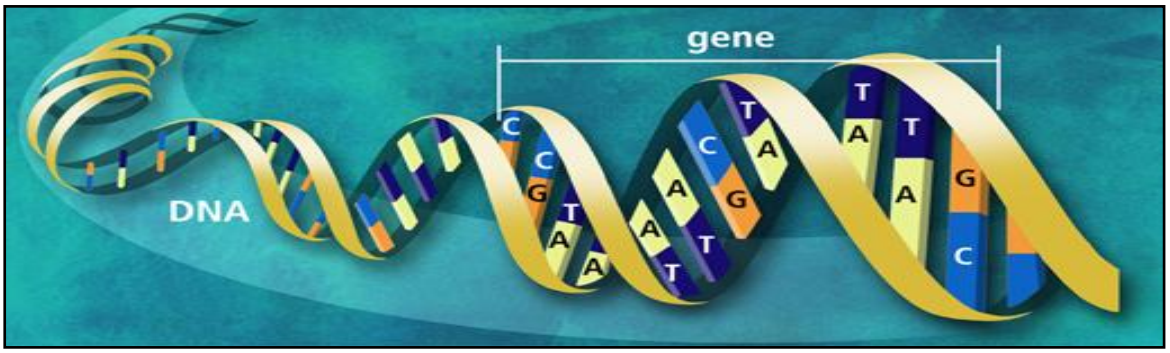


Figura 1-1. ADN humà

A la ***figura 1-1*** podem veure la representació que s'acostuma a donar a l'estructura del *ADN* humà. A nivell general podem veure la cadena d' *ADN* com una **seqüència** llarga *d'informació genètica*. Però si filem més prim podem identificar com aquesta informació genètica és organitzada en forma de **doble hèlix** i està composta per un **ordre concret** de **Nucleòtids**.

Aquests nucleòtids o bases són quatre i són coneguts amb el nom de :

1. **Adenina (A)**
2. **Guanina (G)**
3. **Citosina (C)**
4. **Timina (T)**

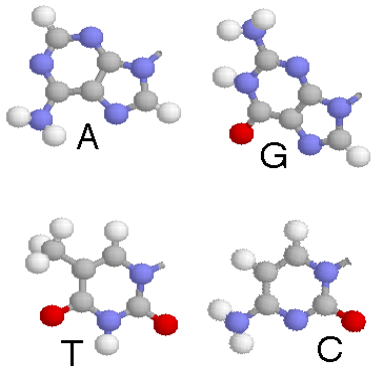


Figura 1-2. Nucleòtids

Ara que ja coneixem que és i de què està composta una seqüència d' *ADN*, només ens manca saber alguns conceptes per a la seva obtenció. Aquest **procés d'extracció** d'*ADN* és fonamental també per entendre en que consistirà aquest *PFC* (Projecte Final de Carrera).

El procés d'extracció del *ADN* es realitza a partir de mostres biològiques com ara la **saliva**, la **sang**, el **cabell** etc. i encara que no ho sembli és tracta d'un procés molt lligat a l' Enginyeria. Aquesta relació existeix, ja que l'extracció constitueix uns processos molt complexes tecnològicament.

Per l'anàlisi i la mostra de resultats s'utilitza un **sistema de seqüenciació** que analitza el material biològic d'entrada. Aquest procés inicialment podia durar dies i inclòs setmanes, però ara està entorn a les cinc o sis hores. La màquina un cop analitza la mostra, imprimeix la seqüència resultant de nucleòtids. La qüestió important està en la manera com s' imprimeix aquests resultats. El problema està en que la tecnologia actual continua tenint algunes restriccions i no permet assegurar amb un 100% de fiabilitat els resultats. Els valors poden tenir errors i ser incomplets i a més poden estar desordenats ja que les mostres d'entrada no guarden cap patró lògic d'organització.

Això és degut a que la tecnologia actual no permet obtenir el genoma humà sencer d'una sola vegada sinó que ho fa mitjançant milers de petits fragments on cada un representa una petita part del genoma. A més aquests petits fragments, també anomenats **short·Reads**, s'obtenen d'una forma desordenada fet que farà necessari un procés posterior d' assemblatge. Aquest procés, anomenat **Re-seqüenciació,** és possible gràcies a la disponibilitat d'un genoma de referència que s'utilitza com a patró.

Es calcula que el genoma humà es composa per varies **Giges** de *bases nucleòtides* (**Gbps**) i per **milions** de *short·Reads*.

Amb aquests conceptes bàsics de Genòmica hauríem de ser capaços ja de distingir les principals problemàtiques que intenta resoldre aquesta ciència juntament amb l'Enginyeria Informàtica. Podem distingir dos grans grups:

1. **Anàlisi de seqüència genòmica:** Implica poder representar i comparar dos o més cadenes d'ADN per tal de buscar similituds i/o diferències rellevants que puguin aportar informació vital sobre les relacions funcionals o evolutives entre les diferents espècies.

2. **Mapeig de fragments curts (short·Reads) d' ADN:** Els algoritmes implementats per aquest tipus de problema s'encarreguen de fer el mapeig dels diferents s**hort·Reads** sobre d'un **genoma de referència**. És a dir, busquen en quina posició del genoma els short·Reads encaixen millor.

1.1 Motivació

El món ha canviat força des dels inicis de la Bioinformàtica i molts han sigut els canvis i avenços en les diferents branques, però molt probablement un dels moments més transcendentals des d'aquell inici, va ser l'aconseguiment del ***Primer esborrani del Genoma Humà***, a càrrec entre d'altres biòlegs, per **John Craig Venter** (considerat el pare del Genoma Humà). Aquest èxit es va informar en una roda de premsa conjunta entre *Tony Blair* i *Bill Clinton* el 26 de juny del 2000. Aquell dia va esdevenir una data crucial de manera que la **biologia**, la **medicina** i la **Bioinformàtica van canviar per sempre**. A partir d'aquell moment es va iniciar una nova etapa al camp de la Genòmica anomedada **Tercera Generació de Seqüenciació.**

Des d'ençà molts **algoritmes Genòmics** han comportat èxits enormes al camp de la biologia i han obert un ventall nou molt gran de possibilitats de cures d' enfermetats. Han obert esperances i una il·lusió de "*medicina a medida*" per cada persona al món.

Un grup del departament **D.A.C.S.O**. dintre del gran projecte *Consolider* **"Supercomputing and e_Science"** del Ministeri de Ciència i Tecnologia, va trobar interessant invertir-hi temps i recursos en un algoritme innovador que resolgués aquest tipus de problemàtica utilitzant els beneficis de la computació paral·lela o **HPC** (High Performance Computing).

1.2 Objectiu Principal

La idea principal d'aquest projecte és la de realitzar un algoritme de **Mapeig de fragments curts d' ADN,** també anomenat com **Read Mapping.**

A més, aprofitant el fort paral·lelisme d'aquest tipus d'algoritmes, ho orientarem per funcionar en un entorn d'execució de múltiples nodes o *cluster* mitjançant el paradigma de programació **MapReduce.**

A nivell general i per començar a endinsar-nos en matèria podem definir el paradigma *MapReduce* com un paradigma introduït per ***Google*** per tractar volums de dades enormes (entorn els *Petabytes)* de forma eficient. Aquest paradigma és un dels més utilitzats per aplicacions intensives de dades ja que proporciona una abstracció al usuari que li permet dissenyar mòduls independentment de la plataforma d'execució. A més també el paradigma resol problemes clàssics molt complexos com ara el balanceig de càrrega, la planificació temporal i espacial i la tolerància a fallades d'entorns distribuïts.

Hadoop és la implementació més popular d'aquest Paradigma i és molt útil per projectes que requereixen **escalabilitat**. Per això i altres raons que més endavant es citaran serà la plataforma escollida per aquest projecte.

Així doncs i a nivell de resum, el nostre objectiu serà l'implementació d'un algoritme paral-lel de Read Mapping utilitzant la plataforma de *Hadoop MapReduce* i la seva *sintonització*.

1.3 Subobjectius

Un cop definits els objectius principals és més fàcil d' especificar els subojectius que són, en la seva gran majoria, conseqüències naturals d'aquests objectius pare.

És important tenir-los presents ja que tot i no ser els objectius principals marcaran en gran mesura l'evolució del projecte i per tant de la mateixa memòria.

Aquests subObjectius són:

- **Realitar un estudi general** sobre els conceptes de Biologia i Genòmica. Aprendre **exercicis bàsics de seqüenciació genòmica : Alineaments** de seqüències d'ADN.

- **Realitar un estudi profund** per tal d'adquirir una bona formació i experiència amb el **framework** MapReduce de **Hadoop.**
- **Realitar un estudi profund** d'algoritmes genòmics de **Read Mapping** i estudiar **l'aplicació** del framework de **Hadoop** en aquest tipus d'aplicacions.
- Estudiar dos algoritmes genòmics importants de Read Mapping : **CloudBurst** i **MAQ**.
- Realitar una **fase de proves exhaustiva** per tal de realitzar una **etapa final de Sintonització i optimització** de l'algoritme**.**
- **Comprobar** el bon funcionament de l'algoritme i **l'escalabilitat** d'aquest al *cluster*.

1.4 Tasques

Tot gran projecte, ja sigui d'enginyeria o de qualsevol altra temàtica, requereix d'una bona planificació per al seu correcte funcionament. Per aquest motiu veiem imprescindible realitzar una planificació inicial de totes les diferents etapes que es poden preveure en un inici per aquest projecte:

- Definició d'Objectius i planificació.
- Estudi de l'estat de l'art dels algoritmes genòmics. *Que fan i per a què s'utilitzen?*
- Analitzar i estudiar el framework MapReduce de Hadoop.
- Elaboració d'un algoritme genòmic nou utilitzant la tecnologia de Hadoop.
- Anàlisis i proves de l'algoritme.
- Fase d'optimització i Sintonització de l'algoritme.
- Generació de la documentació adequada.

Aquests punts representen uns **objectius generals** i poc específics que a priori es poden estimar en el projecte. Evidentment aquests es poden veure modificats i ampliats donada la mateixa evolució del projecte.

En aquesta planificació inicial s' haurien d'afegir també altres tasques de suport que es donen per suposat i que no s'especifiquen: reunions amb el director de projecte,

cerques d'informació per elaborar les diferents etapes del projecte i de la memòria, l'estudi de la extensa documentació passada pel director, instal·lacions del software al cluster etc.

Els objectius principals juntament amb les tasques de suport, ens permeten fer una **planificació més específica** i desglossada de les tasques. Aquesta planificació, molt probablement, ens ajudarà molt en l'evolució del PFC.

- **Setembre 2009- Octubre 2009**

  -Formació i **estudi de** coneixements de **Biologia i Genòmica**.
  -Aprendre **exercicis bàsics d'alineaments** simples i múltiples de seqüències d'ADN.
  -Estudi d'**algoritmes genòmics** de Read Mapping.

- **Octubre 2009- Novembre 2009**

  -Formació i **estudi del framework** MapReduce de **Hadoop**.
  -Buscar l'aplicació entre el paradigma MapReduce i situar-ho amb la problemàtica dels algoritmes genòmics.

- **Novembre 2009 – Gener 2010**

  -Estudi d'algoritmes de Read mapping implementats: **CloudBurst, MAQ**
  -Fer una **primera implementació de l'algoritme** utilitzant el framework de Hadoop.

- **Gener 2010 – Febrer 2010**

  -**Proves i depuració** de la primera implementació.
  -Cerca de possibles optimitzacions.
  -**Comprobar escalabilitat i anàlisis del comportament de l'aplicació**.

- **Febrer 2010 – Abril 2010**

  -**Aplicar optimitzacions** i obtenir una versió definitiva de l'algoritme.
  -**Sintonització** dels settings **per a un Rendiment òptim**.
  -Escriptura de la **memòria**.

- **Abril 2010 – Juny 2010**

  -**Proves** de funcionament i **escalabilitat al cluster**.
  -Escriptura dels **últims canvis a la memòria**.

1.5 Planificació de les Tasques

La planificació de les tasques és una bona forma d'organització i de resolució de Projectes grans que, juntament amb un **diagrama de Gantt,** és torna visual i molt ràpidament llegible per a qualsevol persona.

Amb *Gantt* podem organitzar totes les tasques i subtasques d'una forma ordenada i gràfica, assignant dates específiques i recursos determinats. Això ens serà molt útil sobretot a l'hora de calcular els costos del projecte en *hores de treball* i en el *sistema Hardware* empleat per al seu desenvolupament.

El diagrama s'ha realitzat amb *Microsoft Project Professional 2007* mitjançant l'especificació de les tasques de l'apartat anterior :

Id	Nombre de tarea	Duración	Comienzo	Fin	Nombre
1	**Formació i estudi de coneixements de Biolo**	22 días	02/11/09	01/12/09	Joan
2	Introducció al problema	2 días	02/11/09	03/11/09	Joan
3	Aprendre exercicis bàsics d'alineaments simpl	10 días	04/11/09	17/11/09	Joan
4	Estudi d'algoritmes genòmics de Read Mappin	10 días	18/11/09	01/12/09	Joan
5	**Formació i estudi del framework MapReduce**	12 días	02/12/09	17/12/09	Joan
6	Documentació oficial de Apache Hadoop	5 días	02/12/09	08/12/09	Joan
7	Instalació Plataforma Hadoop 0.18.3	2 días	09/12/09	10/12/09	Joan
8	Prova d'exemple WordCount amb MapReduce	3 días	11/12/09	15/12/09	Joan
9	Buscar l'aplicació entre el paradigma MapRed	2 días	16/12/09	17/12/09	Joan
10	**Estudi d'algoritmes de Read mapping impler**	41 días	18/12/09	12/02/10	Joan
11	Lectura paper CloudBurst	5 días	18/12/09	24/12/09	Joan
12	Lectura paper MAQ	5 días	25/12/09	31/12/09	Joan
13	Estudi de la API de Hadoop 0.18.3 de JAVA	1 día	18/12/09	18/12/09	Joan
14	Fer una primera implementació de l'algoritme	40 días	21/12/09	12/02/10	Joan
15	**Proves i depuració de la primera implement**	26 días	15/02/10	22/03/10	Joan
16	Proves en Single Node	7 días	15/02/10	23/02/10	Joan
17	Proves en HPC	15 días	24/02/10	16/03/10	Joan
18	Comprobar escalabilitat i anàlisis del comport	2 días	17/03/10	18/03/10	Joan
19	Búsqueda de possibles optimitzacions	2 días	19/03/10	22/03/10	Joan
20	Informe Previ	20 horas	15/02/10	17/02/10	Joan
21	**Aplicar optimitzacions i obtenir una versió c**	32 días	23/03/10	05/05/10	Joan
22	Sintonització dels settings per a un Rendiment	15 días	23/03/10	12/04/10	Joan
23	Proves en HPC	15 días	13/04/10	03/05/10	Joan
24	Comprobar escalabilitat i anàlisis del comport	2 días	04/05/10	05/05/10	Joan
25	**Proves a gran escala al cluster**	20 días	06/05/10	02/06/10	Joan
26	Proves en HPC	20 días	06/05/10	02/06/10	Joan
27	Comparació amb algoritmes semblants com M	15 días	06/05/10	26/05/10	Joan
28	**Redacció de la memòria**	80 días	09/02/10	31/05/10	Joan
29	**Revisió director**	86 horas	26/10/09	31/05/10	Porfidio

Figura 1-3 Planificació del Projecte

Podem observar com a la **figura 1-3** s'han comptabilitzat tant les tasques principals com les de suport. Entre elles destaquen per la seva complexitat la primera *implementació de l'algoritme* amb una longitud de **40 dies** , la de *proves*, que en conjunt amb les de *single Node* i *cluster* fan un total de **46 dies** i l'etapa d'*optimitzacions i Sintonització* de **15 dies**.

Tenint en compte que la **dedicació** al projecte es de **4 h/dia**, això fa un total de **160h**, **184h** i **60 h** respectivament només per les tasques descrites anteriorment.

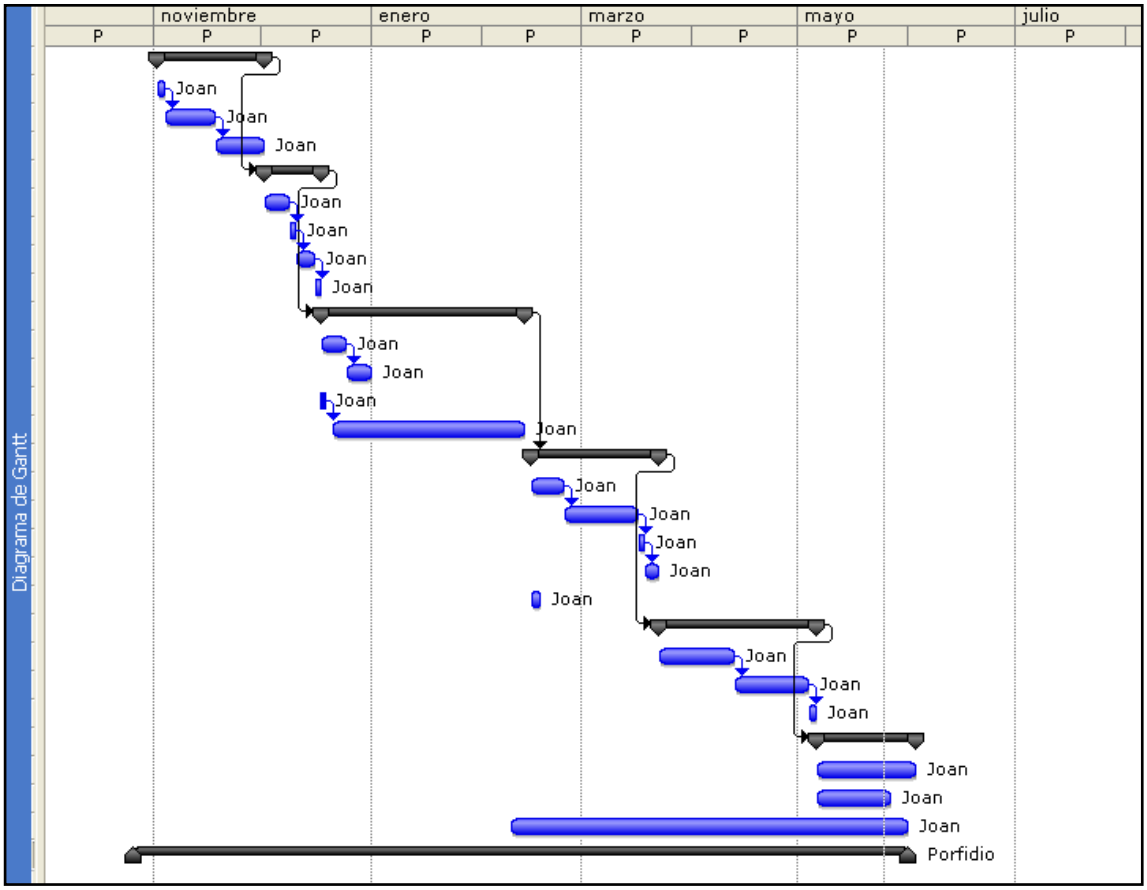


Figura 1-4

Per la **figura 1-4** podem observar que existeixen **2 tasques especials** continues. Aquestes representen la redacció de la memòria al llarg de tota la segona part del projecte i la tasca de revisió realitzada pel director de projecte la qual s'ha estimat en una dedicació de 3 hores setmanals.

Així doncs estem parlant d'un projecte d'un volum de feina en hores de treball de **612 hores** + **65 hores** (memòria) + **86 hores** del director **= 763 hores**.

1.6 Organització de la memòria

Aquesta memòria està dividida en **sis capítols** diferents:

- El **capítol 1,** ja vist anteriorment, introdueix els conceptes necessaris per entendre correctament el projecte i especifica els objectius i subojectius d'aquest.

- El **capítol 2** amplia els conceptes esmentats al primer capítol i descriu l'Estat de l'Art de la Seqüenciació abans i després de la Tercera Generació de Seqüenciació, moment quan apareixen els algoritmes de Read Mapping. També s'explica amb detall el paradigma Map Reduce i la seva implementació més popular: **Hadoop**.

- El **capítol 3** fa un anàlisi de l'Estat de l'Art i discuteix les diferents solucions existents al problema. El disseny, l'altre bloc del capítol, repassa la solució escollida a l'anàlisi i realitza l'estudi de la viabilitat real del Projecte.

- El **capítol 4** tractarà exclusivament de la solució proposada. **L'algoritme i la seva arquitectura** utilitzant la plataforma **de Hadoop.**

- El **capítol 5** forma un *bloc clàssic* que tot PFC *d'enginyeria* hauria de tenir: proves, anàlisi de resultats, anàlisi de prestacions de l' algortime, sintonització etc.

- Finalment el **capítol 6** serà el bloc destinat a les *conclusions* importants del projecte i als factors interessants que ens haguem trobat durant la realització d'aquest.

Capítol II

Fonaments i Estat de l'Art

Per poder comentar *l'Estat de l'Art* de la **Seqüenciació del Genoma Humà** ens és imprescindible primer introduir alguns conceptes bàsics de *Genòmica*. Amb aquests conceptes, se'ns farà més senzill entendre la totalitat d'aquest Projecte i situar-ho en el marc computacional actual.

2.1 Genòmica

En els inicis de la història del ésser humà, el procés d'herència biològica era un pensament remot i inexistent. No hi havia cap mena de teoria ni descobriment al respecte. Només es coneixia, a través de l'experiència, que els descendents d'una parella acostumaven a *"assemblar-se"* als propis progenitors.

A través dels anys es va observar com trets físics presents en algun dels pares, el qual el diferenciaven molt de la resta, acostumava a ser passat directament als fills.

Van haver de passar molts anys fins que científics i personatges importants aconseguissin formalitzar les relacions biològiques i establissin unes teories consistents. Entre ells destaca **Gregor Mendel-Angustuos.**

Figura 2-1 Mendel

Mendel, 1822-1884, va ser un capellà austríac amant de la botànica seguidor de la *teoria evolutiva* de **Charles Darwin.**

Mendel, totalment convençut de que existia una relació estreta entre la mateixa espècie i la pròpia situació familiar, va aconseguir establir a partir d'un senzill experiment les tres lleis fonamentals de la genètica moderna actual.

A més *Mendel* va ser dels primers investigadors genètics en introduir el concepte de **gen**, tot i que ell s'anomenava a aquest com *"elemente"*.

La **Genòmica,** a diferència de la genètica, treballa amb el conjunt de gens de manera integral i com un tot. Aquesta visió molt més recent (*any 2003*), implica forts canvis en la forma d'abordar els problemes però a la vegada obre un ventall de solucions enorme.

Tot i les diferències importants entre les dues branques, Genètica i Genòmica treballen amb la mateixa representació de les dades d'entrada.

Al capítol anterior havíem parlat de *genoma*, *cadenes d'ADN* i fins i tot de *short·Reads* o fragments curts d'ADN. També vam parlar de la forma de doble hèlix del ADN i de la seva composició amb *nucleòtids*.

Així doncs una visió més detallada del ADN seria:

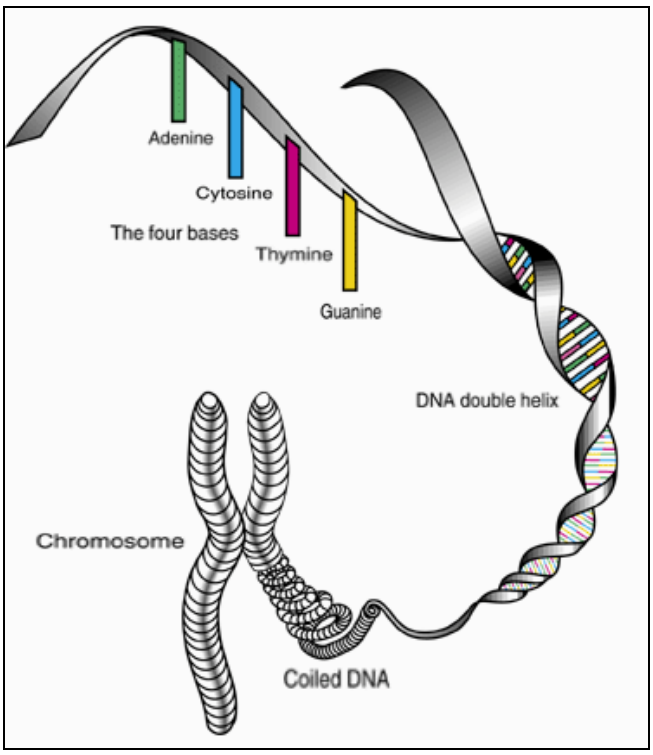


Figura 2-2 Representació ADN

A la *figura 2-2* es pot observar millor la forma de doble hèlix i com tota la seqüència d'ADN està formada literalment per nucleòtids. A la figura, a més, també es pot apreciar un element nou: el **cromosoma.**

Els *cromosomes* són elements molt importants que cal identificar dintre del genoma ja que és tracta d'unitats amb significat propi que identifiquen diferents funcionalitats del genoma.

Cada espècie d'ésser viu disposa d'un número diferent de cromosomes: la mosca en té 8, la ceba té 16 i l'ésser humà en té 23 parells, heretats cada un de la mare i el pare.

El cromosoma humà més conegut per tots és el cromosoma 23: el **cromosoma sexual**. Aquest cromosoma es desenvolupa a les poques hores de la fecundació i determinarà inequívocament el sexe del nou nat. Si es tracta del **parell XX** serà una nena i si és **XY** serà un nen.

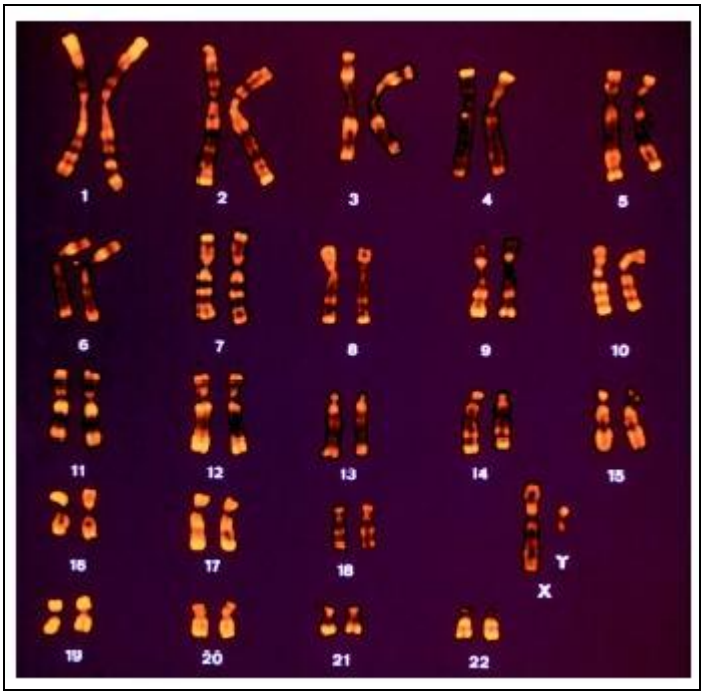


Figura 2-2.2 Imatge Real ADN

Així doncs ara ja sabem que l'ADN no és un text enorme i desordenat de nucleòtids que codifica la nostra informació genètica, sinó que aquesta informació està correctament organitzada i ordenada en diferents cromosomes, que són a la seva vegada fragments molt grans d'ADN.

2.2 Alineament de Seqüències

A l'apartat anterior hem vist com la successió de nucleòtids formen els cromosomes i com aquests a la seva vegada formen el genoma. En aquest apartat però, quan parlem de seqüències, ens referirem únicament a fragments d'ADN amb un propòsit purament teòric i sense un significat propi.

Així, seran seqüències d'ADN **vàlides**:

a) ATATTGCTACGTATATCA

b) ATATATGCTACGTATCAT

c) ACGT

d) A

.
.
.

Alinear significa **comparar**. Per poder alinear seqüències d'ADN i avaluar el seu percentatge de qualitat és necessari prèviament haver-les comparat. Amb un alineament podrem concloure el grau de similitud entre dos subjectes i trobar dominis funcionals comuns o particulars.

Existeixen dos tipus d'alineaments:

1. **Alineament global:** Compara dues o més seqüències cobrint el total de les bases del genoma.
2. **Alineament local:** S'alineen només les parts de les seqüències més semblants.

L'alineament global s'utilitza quan s'està segur que les cadenes a comparar pertanyen a la mateixa espècie i s'espera una similitud molt alta en la totalitat de les seves parts.

L'alineament local en canvi, acostuma a ser la millor opció quan no s'està segur de que les seqüències s'assemblin molt al llarg de tota la seva extensió. De vegades hi ha seqüències que es poden assemblar molt en un punt però divergir molt en d'altres.

El nostre projecte és d'un tipus d'alineament especial anomenat de *Seqüenciació de short·Reads* però es situaria dintre del tipus d'alineament global ja que sempre estarem tractant amb el tamany sencer del genoma.

2.2.1 Exemple d'alineament

Si treballem amb les seqüències d'exemple **A** i **B** tenim :

Alineament simple

```
a:  ATATTGCTACGTATATCA
    ||||
b:  ATATATGCTACGTATCAT
```

4 coincidències

Entre A i B veiem que existeix un percentatge de coincidència molt baix. I encara ho és més al considerar-ho de tipus global. Té un índex de coincidència de **4/18.**

Existeix una tècnica anomenada **Gap** que s'utilitza per simbolitzar un "*forat*" permès en la comparació de dues o més seqüències. Això és degut a que de vegades les màquines d'extracció d'ADN poden cometre errors i no donar una representació genòmica exacta. En aquests casos mitjançant la tècnica de Gap permetrem una flexibilitat major en l'alineament.

Alineament amb Gaps (en A)

```
a: ATAT-TGCTACGTATATCA
   |||| ||||||||||
b: ATATATGCTACGTATCAT
```

14 coincidències

En aquest cas, permetent gaps en A, apreciem com la qualitat de l'alineament ha augmentat molt. Ara parlem d'un índex de coincidència del **14/18**.

Alineament amb Gaps (en A i B)

```
a: ATAT-TGCTACGTATATCAT
   |||| |||||||  ||||||
b: ATATATGCTACG--TATCAT
```

15 coincidències

Finalment si treballem amb la tècnica del Gap en ambdues seqüències l'índex continua millorant. Hem de ser conscients però, que és precisament la **tècnica del Gap** la que fa de **l'alineament de seqüències una operació no trivial**.

2.2.2 Puntuació en alineaments

La puntuació ens els alineaments s'ajusta en base als següents factors:

- Número de bases que coincideixen (Match)
- Número de bases que no coincideixen (Missmatch)
- Número de forats (Gap)

De manera que el sistema de puntuació queda : ***match: +1, mismatch: 0, gap: -1***

Si treballem amb l'exemple al segon cas, tenim que la puntuació és:

Alineament amb Gaps (en A)

```
a: ATAT-TGCTACGTATATCA
   |||| ||||||||||
b: ATATATGCTACGTATCAT
```

puntuació: 14 matches * 1 + 3 missmatch *0 + 1 gaps * -1 = 13

2.3 Tercera Generació de Seqüenciació

Com en la gran majoria de ciències, a mesura que el coneixement humà augmentava, el número de descobriments i èxits a la genètica també creixia. Han sigut moltíssims els progressos que s'han viscut en els últims 50 anys en el camp de la **Seqüenciació** i moltes les tècniques que s'han emprat.

Entre tots els procediments diferents podríem parlar de dos grans grups. Les tècniques inicials de seqüenciació, emprades entre els anys 70 i 90, i les noves tècniques desenvolupades a partir de l'èxit del projecte " **Projecte Genoma Humà**".

El primer grup, més fortament lligat amb tècniques químiques i biològiques molt més complexes, va significar el principi de la Seqüenciació genòmica en busca de la composició total del genoma humà. D'aquest grup podem destacar les tècniques de la **seqüenciació de Maxam Gilbert**, o coneguda com "*seqüenciació química*", i la de **Terminació de Cadena**, o actualment perfeccionada i coneguda amb el nom de **Tècnica de Disesoxi**, elaborat per **Frederik Sanger.**

La tècnica de **Maxam Gilbert,** pionera a la història de la seqüenciació, va resoldre un dels majors reptes a la Genètica al aconseguir seqüenciar un gen sencer del **Bacteriòfag MS2** al 1972. La tècnica estava basada en la **modificació química del ADN** per tal de separar-ho en bases químiques i així determinar l'ordre dels nucleòtids. Per aconseguir-ho es feia necessari prèviament clonar cada començament de lectura del ADN i una sèrie de reaccions químiques.

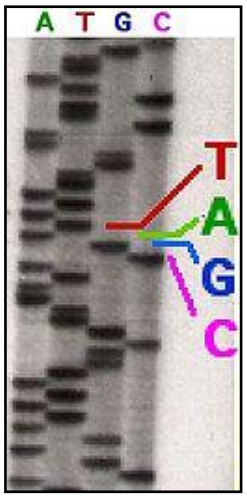


Figura 2-3 Mostra d'ADN amb marcatge radioactiu

Tot i els èxits aconseguits per aquesta tècnica, amb els anys va caure en desús degut a la seva complexitat tècnica, l'ús extensiu dels **productes químics radioactius** perillosos i **les dificultats d'escalar-la** per genomes més grans.

La tècnica de **Terminació de Cadena** era més eficient i utilitzava menys marcatge químic. Ràpidament es va convertir en el mètode d'elecció i va desbancar a la tècnica de *Maxam Gilbert*. La clau va estar en que **Sanger** va treballar amb els **didesoxinucleòtids trifosfats** (*ddNTPs*) com terminacions de la cadena d'ADN.

Tot i així la tècnica de *Disesoxi* tampoc podia seqüenciar directament fragments d'ADN superiors a les 1000 bases/reacció química. Per aquest motiu, la tècnica va continuar avançant i es va evolucionar cap a un intent de **Seqüenciació a gran escala**. **Tabor** de la *Hardvard Medical* i **Carl Fueller** de la *USB biochemicals,* van solucionar als finals dels 70 algunes limitacions introduint tècniques més innovadores.

Tabor i Carl Fueller van saber corregir las limitacions introduïdes col·lateralment pel treball amb els *ddNTPs* i van veure com l'ADN podia ser purificat a partir de *colonies bacterianes*. D'aquesta manera es generaven fragments d'ADN més llargs que més tard s'assemblaven electrònicament en una sola seqüència llarga per un mètode de *força bruta* o "**shotgun**". Aquest mètode no requeria informació preexistent sobre la seqüència de ADN i se la coneix com **Seqüenciació de novo**.

Figura 2-4 GS-FLX 454 Roche

La *Seqüenciació de novo* va ser liderada pels **Laboratoris Roche** i encara és utilitzada a l'actualitat amb el sistema *GS-FLX*. S'ha comprovat com aquesta tècnica pot arribar a ser molt efectiva per genomes grans del quals és desconeix l'origen o espècie animal. Malgrat això aquest tipus de seqüenciació comporta desavantatges importants com ara la utilització de tecnologia molt complexa, cara i amb un marge d'error gran, degut en part, a la redundància natural de l'ADN.

Les possibilitats d'aquesta tecnologia i la visió de negoci per part del sector privat, va fer evolucionar i simplificar aquestes tècniques per tal d'aconseguir una seqüenciació a baix cost.

Aquesta evolució finançada conjuntament per **institucions públiques i privades** van donar lloc a tota una nova tecnologia marcada com hem dit anteriorment pel "**Projecte Genoma Humà".**

El **Projecte Genoma Humà** *(PGH)* és un dels projectes més ambiciosos i grans que ha dut a terme l'home en tota la seva història i va ser dirigit per **James D. Watson** i subvencionat en gran part pels instituts de salut dels **EUA** i el Departament d'Energia.

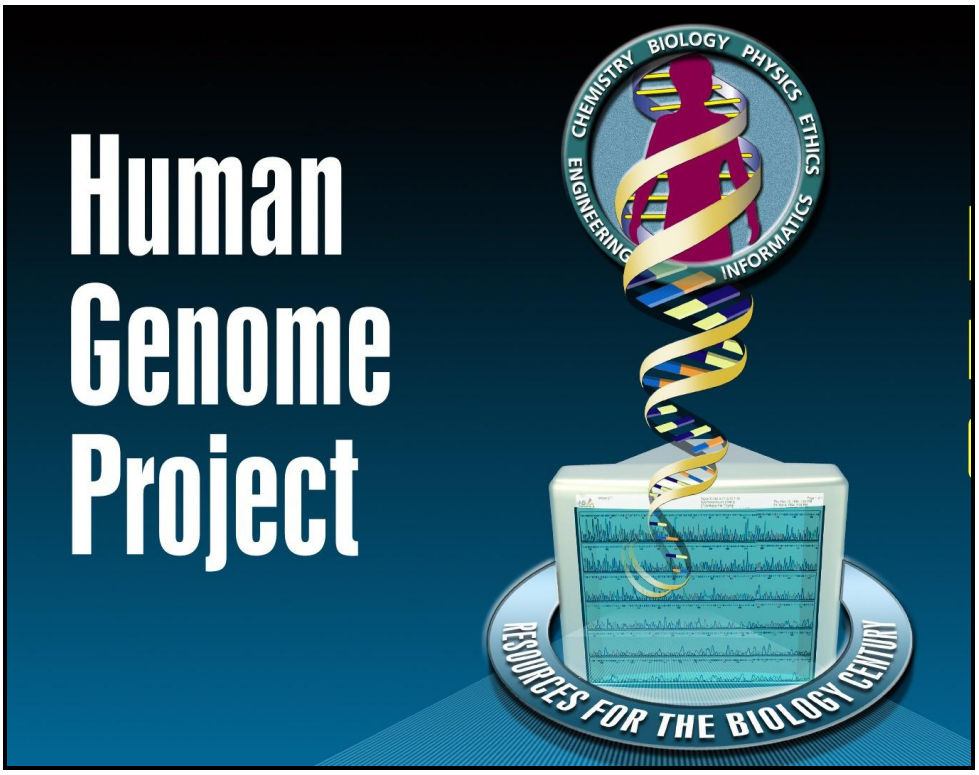


Figura 2-4 Logo del Projecte PGH

Aquest projecte, iniciat al 1990, tenia l'objectiu principal de determinar la **seqüència genòmica completa** del genoma humà (20.000-25.000 gens) ja que fins al moment només es coneixia fragments d'alguns dels seus gens.

El projecte dotat amb 90.000 milions de dòlars va ser estimat amb una llargada de quinze anys però va adquirir tal repercussió mundial, tant pel sector públic com pel sector privat, que va poder ser finalitzat amb èxit dos anys abans del projectat.

En el projecte van participar les millors universitats del món com la de Califòrnia dels EUA, Canadà, Nova Zelanda i Gran Bretanya.

Dos anys abans de la finalització del *PGH*, **l'any 2001**, es va publicar un **esborrani molt aproximat** del genoma humà complet en una important roda de premsa. Aquesta roda de premsa, com ja vam fer menció al primer capítol, va marcar un **abans i un després** al món de la Genètica i per la Genòmica, recentment "nascuda".

El resultat final del projecte va deixar xifres com les següents:

- El genoma humà complert té **38.000 gens**. Això implica que només té 1/3 de gens més que el cuc comú, i només 5.000 més que la planta *Arabidopsis*.
- El genoma humà té entorn les **3,3 G*bases*** i es calcula que és com a mínim un **98% idèntic al** del **ximpanzé** i altres primats.
- Es van utilitzar tres dones y dos homes per seqüenciar el genoma humà : Un afroamericà, un xinès, un asiàtic, un hispanoamericà i un caucàsic. Cada un compartia més d'un **99.99%** del mateix **codi genètic**.
- S'estima que més d'un 35% del genoma conté seqüències repetides. Aquests fragments se'ls coneix com **ADN brossa**.
- Actualment existeix una **versió en paper** del genoma humà sencer al **museu Wellcome Collection** de Londres. Aquest ocupa més d'un centenar de toms, cada un d'ells amb més de mil pàgines amb una lletra de tamany petit.

Figura 2-5. Museu Wellcome Collection

Amb aquestes xifres i després del èxit causat pel *PGH*, no és d'estranyar doncs que es produís un autèntic "**Boom"** per part del **sector privat** orientat al món **genòmic.**

Les companyies mèdiques i farmacèutiques més prestigioses del món van començar a oferir una oferta sense precedents: Un *genoma a la carta a un preu competitiu*. Empreses com **Pacific Biosciences** o **Solexa-llumina** asseguraven poder seqüenciar el genoma humà sencer per poc més de 5000 $ i en menys d'una hora de computació. Un preu ridícul si tenim en compte la complexitat d'aquestes operacions i el profit que es pot treure d'elles.

En l'actualitat aquest camp està esdevenint tan competitiu i innovador que ja se l'anomena com una generació nova de seqüenciació: la **Tercera Generació de Seqüenciació**, o *Next Sequencing Generation*.

Actualment s'està arribant a límits on ara fa anys era només fantasia. Per aquest motiu es difícil predir a on podrem arribar en tan sols deu anys. Altres governs com el de Xina, amb l' institut de Genòmica de Pekín (**Beijing Genomics Institute**) van encara més enllà i asseguren obtenir un resultat amb una qualitat igual de bona que el de Biosciences però baixant el preu a 1000$ i a poc més de 3 minuts d'execució.

És clar doncs, que la guerra entre empreses està servida. Donant lloc a una vertadera nova generació de seqüenciació, on **l'Enginyeria Informàtica** juga un **paper fonamental.**

2.4 Algoritmes de Mapeig de Fragments Curts

En els anys posteriors a l'any 2000 i amb la majoria d'esforços centrats al genoma humà, es va començar a parlar més amb el terme de *Genòmica* que amb el de *Genètica.* A més, amb un patró fidedigne del genoma humà, ara si semblava possible simplificar els mètodes de Seqüenciació.

Companyies com *Solexa-Illumina*, *454* de Roche, *Pacific Biosciences* etc. van començar a implementar tècniques molt innovadores de seqüenciació i a la vegada molt senzilles que anaven, altre cop, molt properament relacionades amb l'Enginyeria Informàtica.

Va ser en aquell moment quan van començar aparèixer tota una sèrie d'algoritmes informàtics que es coneixen amb el nom d'Algoritmes de Mapeig de Fragments Curts, o **Algoritmes de Read Mapping**.

Aquest algoritmes, com veurem en aquest capítol amb detall, tenen dues particularitats importants:

- **El gran tamany de les dades d'entrada.**
- **El fort paral·lelisme aplicable a la naturalesa del codi.**

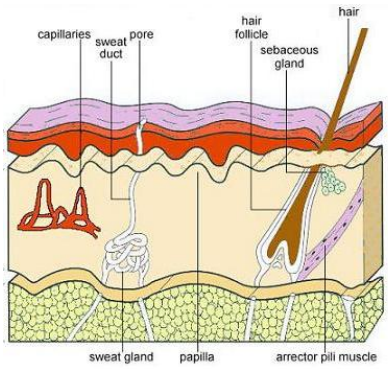
Malgrat que el punt de vista d'Enginyer Informàtic s'hauria de centrar només en la resolució del problema i en l'estudi dels algoritmes, trobem convenient fer una petita explicació de com són i com s'obtenen les dades d'entrada. D'aquesta manera resultarà més senzill entendre les particularitats d'aquest tipus d'aplicacions.

▪ Procés D'extracció

El procés del *Read Mapping* comença amb una extracció de **mostra d'ADN**. Aquesta mostra pot ser saliva, sang, cabell humà etc. A continuació la mostra és introduïda en un **seqüenciador** el qual l'analitzarà.

Durant la **Tercera Generació de Seqüenciació** han aparegut nombrosos seqüenciadors diferents. Tots ells amb particularitats diferents: temps de càlcul, índex d'error en l'anàlisi, precisió en els resultats de sortida etc. En el nostre cas, al treballar amb dades ja generades, només ens importarà l'últim factor.

A nivell d'esquema podríem veure aquest procés com el de la **figura 2-5** :



Les mostren són analitzades pel Seqüenciador i a continuació s'imprimeixen els resultats en fitxers de Text.

Genome Sequencer Polonator, Hardvard

RESULTS

Figura 2-5. Esquema extracció Read Mapping

El **seqüenciador Polonator** de la universitat de Hardvard és capaç d'imprimir **400.000 short·Reads** amb una longitud de fins a **250 bps** amb un índex d'encert del **99,5%.** Aquests **short·Reads** són impresos en un fitxer de sortida especificat per l'usuari i tenen un **format FASTA.**

El format *FASTA* és molt semblant al ja esmentat durant tota la memòria : seqüències de nucleòtids amb un ordre determinat amb l'afegit d'un identificatiu (normalment numèric) per cada Short·Read.

Short Reads

```
>1
GCCTGTTCTTTACATGATTTTTGGTCTAGTGTATGG
>2
AACCGCTGTAAAGGCTTCTGCCACACCGATTTCTTG
>3
GAGGTGATTGTGGTATTGT.GGTAAATCGGTGATTG
>4
GCTTTAGCCGACCTGAACT.GACTACAAGTTGACCA
>5
AAAGGCTACCCGCGGTTGAACCTTACGTGACACATT
>6
GCTGTGGGGGTGGTTGGGTCTTGTTTGGTTTGTGCC
>7
GGACTATTTGAAGTTGTCA.CAAGAAAAATCAAAGT
>8
ATAAGCATCAATACCACTAATATAGATATTAAAGCT
>9
TCCTCGCTTAATTCAAAAAGTTGTTGGGTTAAGTCT
>10
ACCAACACGAGCTGGTTCAACTTGATAATTGACTTT
>11
GATTACTTGTAATTATACGTTACAAAATCATATTGG
```

Figura 2-6. Exemple Fitxer Resultat

Els fitxers de sortida poden tenir perfectament un tamany superior als 100 MB i es caracteritzen per tenir una longitud fixa per totes les mostres short·Reads. A la **figura 2-6** aquestes mostres tenen un tamany de **36 bps**. Aquest factor, com ja hem dit, dependrà del seqüenciador utilitzat.

És important saber que el fitxer de sortida malgrat tenir un identificatiu únic per cada mostra short·Read no guarda cap ordre concret. Això és degut a que la tecnologia de *Tercera Generació de Seqüenciació* ha simplificat molt el procés per tal d'abaratir costos. Per aquest motiu seran necessaris una sèrie de procediments intermitjos abans d'obtenir el genoma resultat. Un d'aquests procediments és el procés de Mapeig.

▪ Procés De Mapeig

Els algoritmes de *Read Mapping* tal i com indica el seu nom comencen just en aquest punt. **"Mapejar"** significa trobar una **correspondència** entre dos punts, una **associació**. És precisament aquest concepte el que tracten de solucionar els algoritmes de *Read Mapping*.

Com comentàvem al apartat anterior, gràcies al èxit de **Projecte Genoma Humà**, desde l'any 2003 disposem del genoma humà complert. A més, també sabem que aproximadament més del 99% del codi genètic entre dos persones coincideix.

Es d'esperar doncs que el **mapeig** de les **mostres d'ADN** extretes es mapegin **contra** aquest **genoma de referència**. Això és així ja que com hem explicat abans les mostres no guarden cap ordre i necessiten d'una fase d'ordenació per poder *"muntar"* el genoma resultant.

Lògicament, tractant-se d'una operació de mapeig, les dades del genoma de referència també s'expressaran en el mateix format que les mostres: el format *FASTA*.

Genoma

```
>Streptococcus_suis
atgaaccaagaacaacttttttggcaacgatttattgaattggcaaaggtaaattttaag
ccatctatttatgatttttatgtcgctgatgcaaaattactcggaatcaaccagcaagtt
gccaatattttcttaaatcgtccatttaaaaaagatttctgggaaaaaaacttcgaagag
ttaatgattgccgctagttttgaaagctacggagagcctcttaccatccaatatcaattt
acagaggatgaacaggagattaggaatactacaaacacaagaagttcaatagttcaccag
gtacagacacttgagccggctactcctcaagaaacttttaaaccggttcattctgatata
aaatcccagtacacctttgctaattttgtacaaggagacaataatcactgggcaaaggct
gcagctttagctgtatctgataacctaggtgagctctacaatccattattcatttttggt
ggtcctggtcttggaaaaactcatattttaaatgcgattggaaataaggttctagccgat
aatccccaggcaaggataaaatatgtctcatcggaaacattcatcaatgaatttttagaa
cacctccgtctcaatgatatggaaagtttcaaaaaaacctatcgcaatctggacttactt
ctaattgatgacattcagtctctccgtaataaagcaacaacacaggaagaatttttccat
acttttaatgcgcttcatgaaaaaaataagcagattgtactcacaagcgaccgtaatccc
gatcacttagacaatttggaagaaagactagtaacacgtttcaaatgggggttaaccagt
gaaatcactccacctgattttgaaacacgtatcgcaattttacgtaacaagtgcgagaac
ctgccttacaactttacaaatgagacgctatcctatctagctgggcaatttgattcgaac
```

Figura 2-5. Fitxer exemple genoma Streptococcus

Fent números ens adonarem fàcilment del perquè de les dues característiques d'aquest tipus d'aplicacions:

1. **El gran tamany de les dades d'entrada.**

   Aquesta característica és directa si tenim en compte que parlem per una banda d'un volum de mostres entorn els **400.000 short·Reads** de **250 bps** *(≈150 Mbytes)* i del tamany del genoma humà complert de **3.3 Gbps** (≈ 4Gigabytes).

   A més si tenim en compte que es poden estar alineant mostres de diferents persones simultàniament veurem com aquests números es multipliquen ràpidament.

2. **El fort paral·lelisme del codi**

   La segona característica també es prou fàcil de trobar-la si tenim en compte que es pot interpretar com si d'una operació d'alineament múltiple es tractés, on la multiplicitat la marcarà el número de mostres d'entrada. És tracta doncs, amb cada mostra, determinar quina és la posició del genoma on es dona un alineament perfecte, o el millor, amb ella. Això si tenim en compte que és tracta sempre del mateix genoma de referència i que els short·Reads entre ells es poden solapar i inclús repetir, veurem com el paral·lelisme és evident.

És per aquests aspectes que l'Enginyeria Informàtica juga un paper tan proper a la branca de la genòmica. No només per la part Software sinó també en l'aspecte Hardware d'altes prestacions com són els **clusters** o els **High Performance Computing.** Unint totes dues branques, s'intenta aprofitar el màxim de paral·lelisme de les dades per aprofitar-ho a nivell de Hardware.

Està clar doncs que amb aquests números i les característiques ben analitzades, una aplicació que sàpiga explotar aquestes característiques molt probablement funcionarà amb un rendiment més que bo.

Tot aquest procés de *Read Mapping* a nivell d'esquema seria el següent:

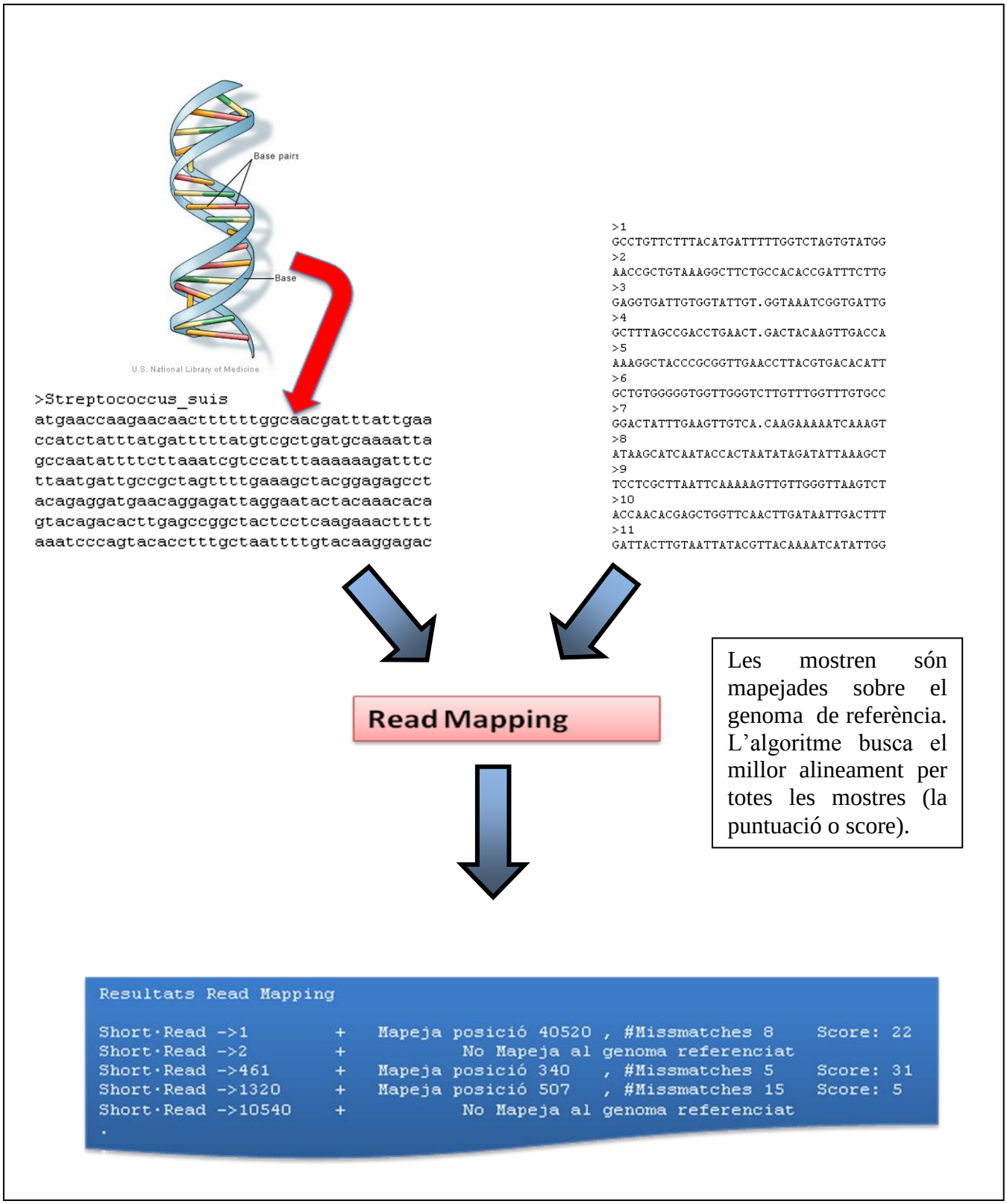


Figura 2-6. Esquema Read Mapping

2.4.1 MAQ

L'algoritme **MAQ** (el link del seu paper està afegit a l'apartat de referències) és potser, sinó ho és ja, un dels algoritmes de *Read Mapping* més important i amb més experiència al món de la bioinformàtica. La seva simplicitat i el seus bons resultats el fan un referent per tots els programadors de la *Tercera Generació de Seqüenciació.*

És tracta d'un projecte realitzat per **Heng Li** i anteriorment l'algoritme es coneixia amb el nom de **mapass2.** *Heng* és l' autor d'altres algoritmes bioinformàtics importants com **BWA, TreeFam etc.** Heng Li ha treballat en nombroses ocasions pel **Beijing Genomics Institute**, un dels centres més prestigiosos del món en recerca genòmica (citat en el capítol anterior).

MAQ es situa com un algoritme de Read Mapping de *Tercera Generació* especialitzat amb les mostres produïdes pel seqüenciador de **Solexa-Illumina** *1G Genetic Analyzer.* Tot i així amb el temps ha sigut adaptat per la comunitat d'usuaris a tota classe de mostres i s'han desenvolupat utilitats extres molt interessants.

A continuació mostrem el **work Flow** de l'aplicació:

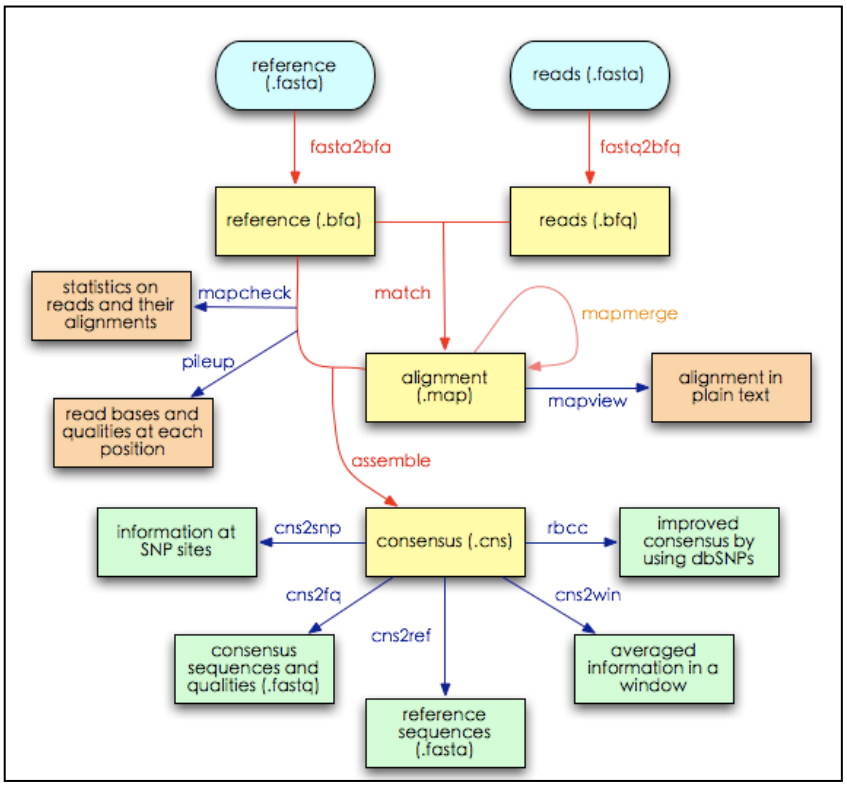


Figura 2-7. Fluxe programa MAQ

Tot i que a primera vista pugui semblar un flux de treball un xic complex no ho és tant si és desglossa en diferents etapes.

- **Etapa de Mapeig**

A la figura podem apreciar com les entrades de l'aplicació són les típiques i ja estudiades al bloc anterior:

-El genoma de referència *(format FASTA)*

-Les mostres short·Reads de Solexa-Illumina *(format FASTA)*

Aquestes entrades, després d'un procés previ d'optimització en la seva representació, són passades a la etapa simbolitzada a l'esquema com **match i alignment.** Aquesta fase és el pas principal de tots els algoritmes de *Tercera Generació de Seqüenciació* : la fase de **mapeig.** Com ja vam comentar al bloc anterior, en aquesta etapa és busca l'alineament de totes les mostres per separat sobre un genoma de referència, en el nostre cas l'humà.

És important saber que *MAQ* treballa amb la tècnica del **GAP**, és a dir, permet un cert marge de flexibilitat a l'hora del mapeig. Aquesta funcionalitat, opcional per part de l'usuari, permet treballar amb un número fixa de gaps: **un, dos** o **màxim tres**. És evident veure que quant més gaps escollim, major serà la complexitat de l'algoritme.

Per aconseguir aquest marge d'alineament es segueix un algoritme molt simple i útil. Trobem interessant explicar amb detall aquesta tècnica ja que molt segurament ens serà de gran ajuda en la implementació del nostre propi algoritme de Read Mapping.

La tècnica és senzilla des del punt de vista tècnic però requereix d'un esforç conceptual per situar-ho en el nostre problema.

Es defineixen **sis plantilles** en tres grups de dos:

```
a)    11110000 i 00001111
b)    11001100 i 00110011
c)    10101010 i 01010101
```

Aquestes plantilles o *templates (*amb una funció lògica *AND* i longitud *en bits* **=** *longitud seqüència d'entrada)* es fan servir per **"filtrar"** les mostres short·Reads d'entrada. Per aconseguir-ho, el que farem serà generar noves mostres a partir de l'original. Per exemple, considerant que volguéssim permetre short·Reads amb un marge d'error de 3 *Bases*, la complexitat de la entrada considerant que tenim ***N*** **mostres** passaria a ser de **6N**. És a dir, una relació de **N ->6N.**

Veiem-ho millor amb un **exemple:**

Considerem un fitxer d'entrada de mostres short·Reads com els habituals que ja hem vist. De totes les mostres d'aquest fitxer (recordem que poden haver més de 400.000) pensem en un sol registre. Aquest registre o short·Read podria ser:

```
>1
AACCGCTGTAAAGGCTTCTGCCACACCGATTTCTTG
```

El registre, en format *FASTA,* té una longitud de 36 bps. Aleshores considerant que l'usuari tria l'opció de **GAP** seleccionant el número màxim (3 *gaps)* i que a més selecciona el paràmetre de *ReadLength* a *28 bps* aleshores el que estaria fent en realitat l'algoritme és:

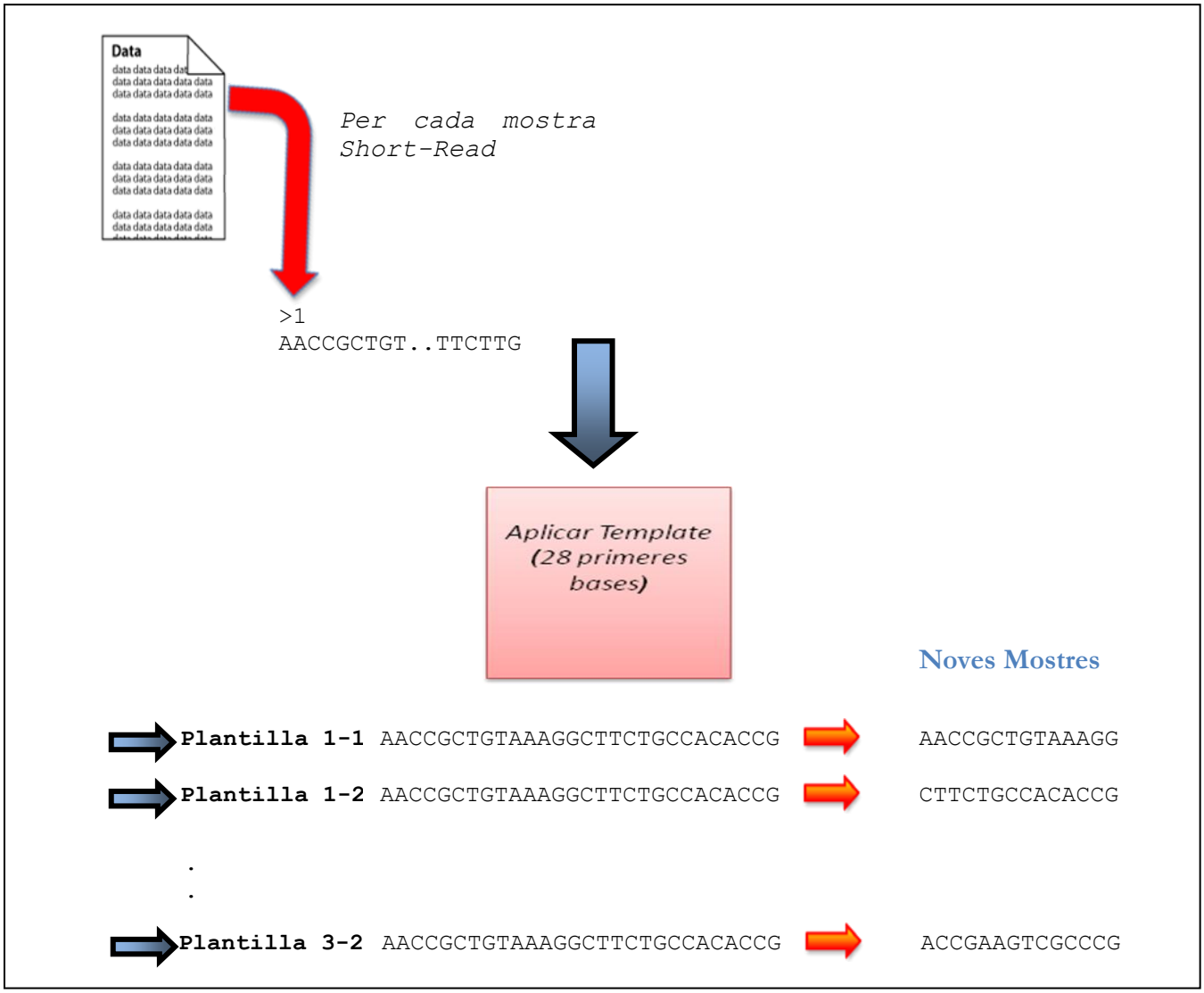


Figura 2-8. Esquema del mapeig amb Gaps

Com veiem a la **figura 2-8** l'algoritme amb la tècnica del *Gap* construeix sis noves mostres short·Reads a partir de la mostra original. Per aquest cas l'algoritme treballa amb les 28 primeres bases (està demostrat biològicament que les parts finals d'algunes mostres acostumen a tenir un índex major d'error en l'extracció) a partir de les quals MAQ genera les **llavors** de longitud *ReadLength/2,* és a dir, 14 bps. Aquestes **llavors,** tot i tenir representacions diferents, seran identificades per igual ja que totes han sigut construïdes arrel d'una mostra original.

D'aquesta manera, aplicant una tècnica algebraica senzilla, estem incrementant el número de mostres d'entrada i així fer que l'índex de probabilitat de mapeig augmenti.

- **Etapa de Consens o estadística**

Aquest pas, tot i ser anomenat diferent, també l'hem comentat anteriorment. És tracta de la fase de puntuació dels short·Reads sobre el genoma de referència. Trobem important estudiar també a fons aquesta tècnica de *MAQ* ja que resulta molt interessant i perquè gran part dels algoritmes de Read Mapping també la utilitzen.

La tècnica de puntuació d'alineament explicada al bloc anterior era correcta però molt general. *MAQ*, al igual que altres algoritmes de mapeig, utilitzen una tècnica per a la puntuació anomenada de **llavor** i extensió o **seed and extend**.

La idea d'aquesta tècnica, tal i com indica el seu nom, és la de treballar amb una **"llavor"** o part minimitzada d'una mostra per més tard **entendre-la** mica en mica per comprovar el seu rendiment.

En el nostre cas la **llavor** seria el fragment de short·Read generat a la etapa de mapeig (recordem que era de *14 bps*) i constituirà l'etapa de *seed* pròpiament. Amb l'etapa de seed és fàcil observar també com a més d'augmentar les possibilitats de mapeig és simplifica també el problema ja que no s'ha de buscar un matching dels registres sencers.

Després de la fase de seed, considerant que la llavor **mapeja** el genoma, entraríem a la fase d'**extend**. En aquesta fase l'algoritme comprovaria la resta del short·Read i es compararia amb el genoma, calculant així la puntuació de l'alineament final obtinguda mitjançant el número de *missmatches* calculats.

És important remarcar un **handicap important** de MAQ. L'algoritme, programat ja fa varios anys, va ser pensat per utilitzar-se en un entorn de desenvolupament de **node únic**. És a dir, amb l'estructura original del codi no es vàlida l'execució en un entorn de còmput paral·lel com el dels **HPC**. Això és una carència important ja que com hem pogut analitzar al bloc anterior, estaríem desaprofitant una de les característiques crítiques d'aquest model d'aplicació: **el paral·lelisme**.

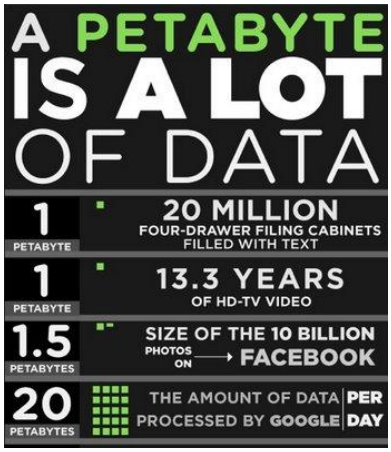
Aquesta carència, comú en pràcticament totes les aplicacions de *Read Mapping* actuals, marca significativament l'estat de l'Art de la *Tercera Generació de Seqüenciació* i és tracta d'una de les raons principals per a la realització d'aquest Projecte.

El projecte MAQ, així com totes les utilitats desenvolupades per la comunitat d'usuaris, és open Source i es pot aconseguir fàcilment des del host de ***SourceForge.net.***

2.5 Paradigma Map Reduce

- **Què és Map Reduce?**

És un paradigma de programació paral·lela orientat per treballar en entorn de *clusters grans*. Aquest paradigma ja àmpliament conegut en altres llenguatges de programació com **Lisp** o **Haskell,** treballa amb un tipus de **programació funcional** amb parelles de **clau-valor**.



MapReduce està dissenyat per treballar amb volums de dades entorn els ***terabytes*** i fins i tot ***petabytes***. Les empreses informàtiques més importants en l'actualitat com ***Google, Facebook i Twitter*** treballen amb aquest paradigma ja desde fa alguns anys.

La seva simplicitat a l'hora de programar i les avantatges oferides pel paradigma, el fan una eina molt potent pels desenvolupadors especialitzats que busquen escalabilitat i un rendiment eficient.

- **Per a què s'utilitza?**

Com ja hem anunciat anteriorment, *MapReduce* esdevé una eina ideal per aplicacions que treballen amb un tamany de dades enorme. Aquest tipus d'aplicacions, anomenades **Aplicacions Intensives de Dades** o *Data Intensive*, es caracteritzen per necessitar un disseny d'algoritmes on l'arquitectura és marcada pel volum de dades.

És per aquest motiu que empreses com *Google* o *Facebook* amb autèntics tamanys de dades descomunals, han apostat per aquest paradigma de programació i l'han anat

perfeccionant amb el pas dels anys (el pioner i creador del paradigma va ser *Google*. Més tard va aparèixer **Hadoop** com inspiració d'aquest).

Nosaltres, amb unes necessitats a priori força similars a aquestes (en menor mesura), i amb els coneixements necessaris d'Enginyeria informàtica, trobem de MapReduce una eina que podria esdevenir molt útil i potent per les nostres necessitats. A més, estaríem entrant en un terreny innovador i interessant ja que fins el moment no s'ha aplicat gaire aquest paradigma al camp genòmic.

▪ Com s'utilitza?

Una de les avantatges importants del paradigma és la simplicitat a l'hora de treballar amb ell. L'usuari només necessita implementar dues funcions importants:

1. **Funció Map**
2. **Funció Reduce**

La resta d'operacions complicada i extremadament sensible és resolta per les pròpies llibreries de *MapReduce*. Tasques com:

- Paral·lelització
- Tolerància a fallades
- Distribució de les dades
- Sistema de Fitxers distribuït
- Balanceig de la càrrega

Si bé és veritat que el model facilita molt el treball amb el tipus d'aplicacions Intensives de Dades, també s'ha de dir que requereix d'una fase important d'anàlisi i sintonització exhaustiva. A més també es fa necessari un estudi previ del tipus de programació i de les característiques funcionals d'aquest.

2.5.1 Detalls de l'arquitectura

El paradigma **MapReduce** utilitza el sistema distribuït per dividir les aplicacions en **tasques independents** més petites. Al tractar-se d'un llenguatge funcional és fàcilment paral·lelitzable ja que els *lp's* (llenguatges de programació) funcionals són per naturalesa concurrents. D'aquesta manera resulta molt senzill escalar els algoritmes per diferent número de nodes i obtenir un rendiment eficient propi dels **HPC**.

A nivell global en tot treball llançat amb el **paradigma Map Reduce** s'identifiquen dues etapes diferents: **Map i Reduce**. Aquesta última però, engloba dues altres fases prèvies : **Shuffle** i **Sort**. Cada etapa o fase es caracteritza per unes funcionalitats molt determinades i una arquitectura molt clara. A la **figura 2-9** veiem l'esquema general del paradigma:

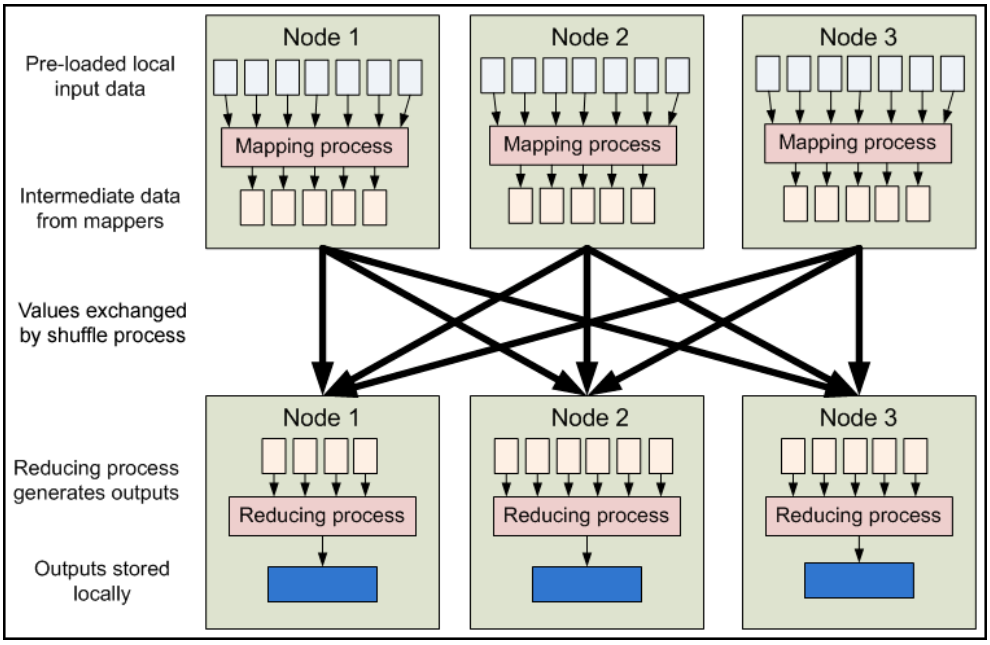


Figura 2-9. Esquema General del paradigma Map Reduce

La figura ens mostra la simplicitat del paradigma Map Reduce. Cada node del cluster executa un número determinat de Maps i de Reduces i **és comunica solament en un punt de l'execució** del treball: a la fase de Shuffle. És important observar com les dades d'entrada del Map, localitzades a l'esquema com "*Pre-loaded local input data*", fan referència a les dades que estan emmagatzemades al **DFS**, és a dir, al Sistema de fitxers Distribuït.

Les dades generades pels Maps de cada node (recordem tuples de forma *(clau,valor)*) són **agrupades per clau** en un procés intermig identificat a l'esquema com "*values exchanged by shuffle process*" i identifica a la fase de Shuffle. És en aquest punt solament,

com anunciàvem línies amunt, on es comuniquen entre si tots els nodes i quan realitzen l'intercanvi d'informació de manera que cada node processa grups de tuples sencers i diferents (cada grup amb una clau diferent).

Finalment el Reduce executarà les funcions pertinents a cada grup i imprimirà els resultats en un fitxer de sortida.

A continuació analitzarem en detall cada fase:

- **Funció MAP**

La funció *Map* amb domini *Map (k1,v1) →list(k2,v2)* treballa amb dades estructurades de tuples *(clau,valor)*. La funció rep com entrada el conjunt total de tuples *(c,v)* i executa la funció Map per cada element del conjunt. Aquesta execució produeix una sèrie de resultats intermedis formats en llista de tuples *(c,v)*. A la **figura 2-10** veiem aquesta relació; el Map, a partir de cada registre d'entrada, construeix una nova sortida (una nova tupla) amb la forma determinada pel programador (una clau i un valor específic).

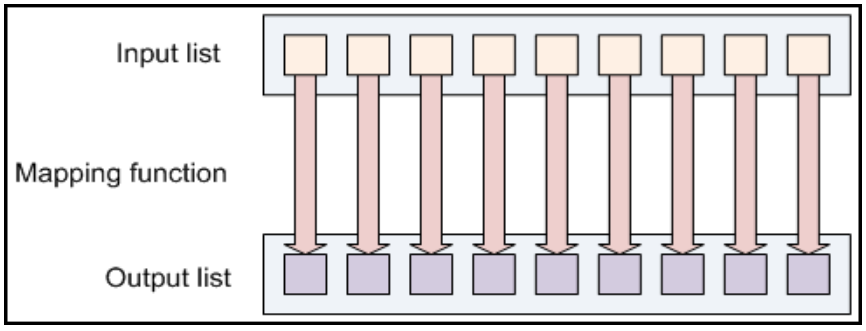


Figura 2-10. Esquema Funció Map

- **Funció d'agrupament i ordenació**

Aquesta funció, **ja definida** per defecte al paradigma, s'encarrega de, un cop finalitzada la fase de *Map*, **agrupar** tots els resultats intermedis per *clau*. D'aquesta manera es generaran noves llistes que a la vegada formaran grups amb un domini comú. Aquesta funció, també anomenada de **Shuffle & Sort**, s'identifica més bé amb una fase en si i és pot veure com una abstracció de filtre amb el domini de: *list(k2,v2) →list(k3,list (vn))*

La **figura 2-11** mostra la relació de **n->1** de la fase de Shuffle. Recordem que aquest agrupament vindrà donat pel valor de les claus (k2).

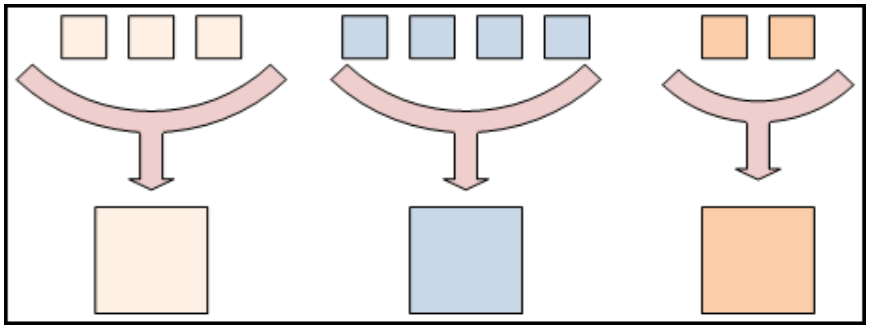


Figura 2-11. Esquema Funció Shuffle & Sort

- **Funció Reduce**

La funció *Reduce* rep com entrada els diferents conjunts agrupats i ordenats per la fase de *Shuffle & Sort* i genera un únic resultat expressat com una llista de valors (únicament valors sense claus).

Per tant el domini del *Reduce és :* $(k3,list(vn)) \rightarrow list(vn)$ on l'entrada més simplificada és de la forma *(clau, [resultat1, resultat2,...resultatN]*. Veiem-ho a la **figura 2-12**.

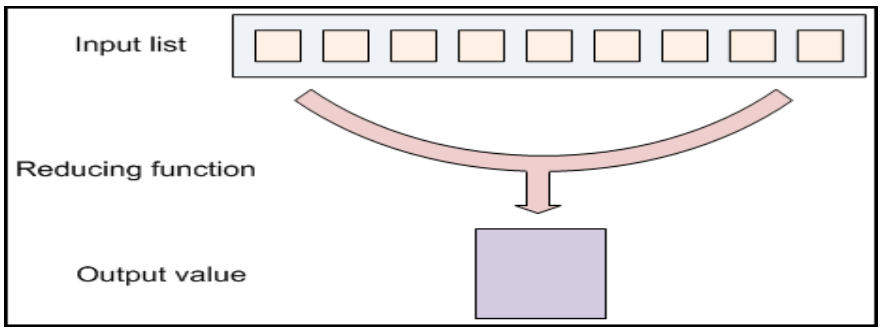


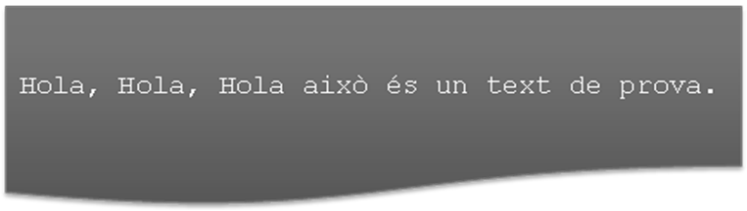
Figura 2-12. Esquema Funció Reduce

2.5.2 Algoritme WordCount

Un exemple molt clàssic per avaluar el *Rendiment* de les aplicacions en entorns paral·lelitzables com ara **MPI** o altres plataformes és l'exemple del comptador de paraules o WordCount. Aquest exemple, des del punt de vista d'enginyer, és molt fàcil de comprendre tant desde l' aspecte funcional com del concurrent.

L'algoritme *WordCount* compta el número d' ocurrències de cada paraula en un text. És immediat veure la paral·lelització d'aquest problema si és té compte que el text està expressat en un format ASCI plà i per tant és divisible sense cap problema en *N* fragments.

Per veure-ho més clar analitzarem etapa a etapa l'algoritme amb un text d'entrada bàsic, suposant el número de Maps i de Reduces igual a un:



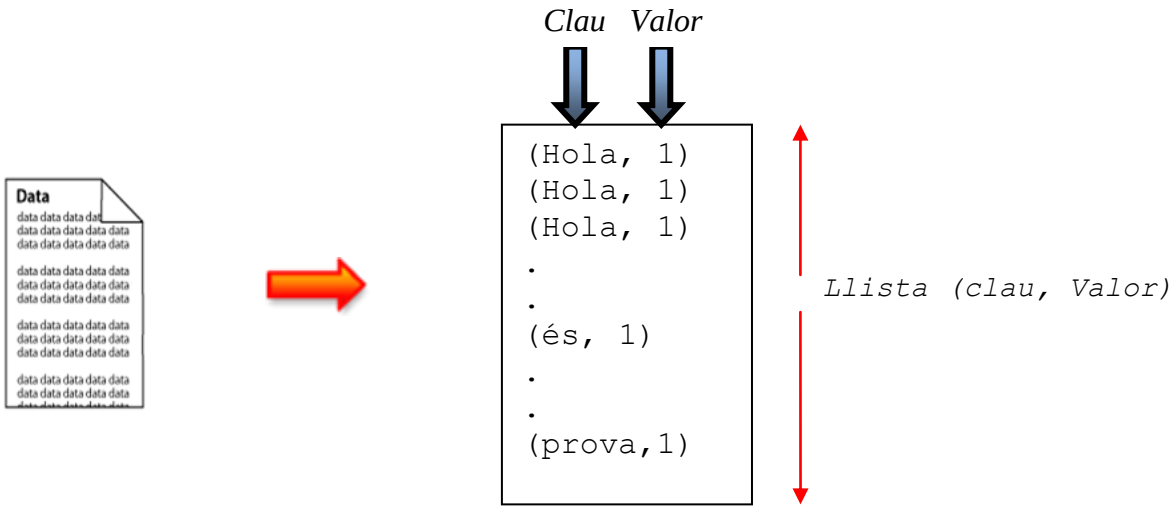
Com podem predir l'algoritme hauria de donar com a sortida:

```
Hola, 3
això, 1
és,   1
un,   1
text, 1
de,   1
prova,1
```

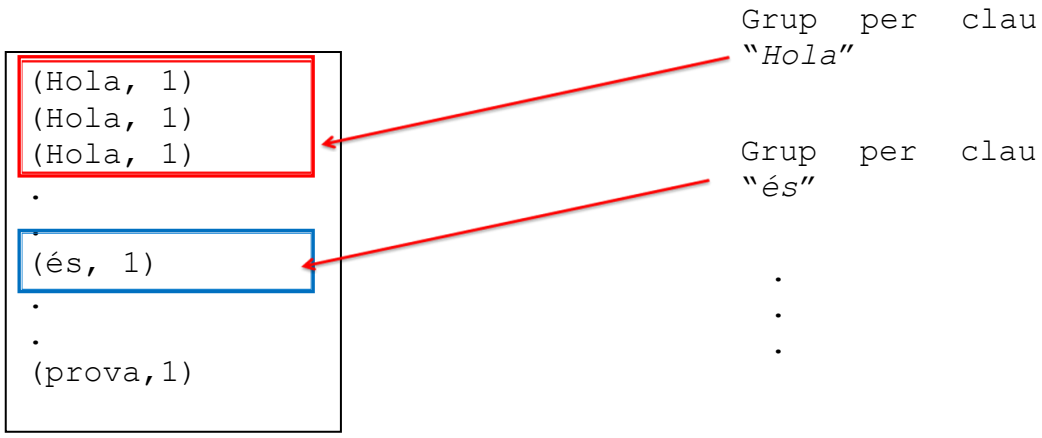
Analitzant-ho per etapes:

- **Map** *Map (k1,v1) →list(k2,v2)*

```
map(String name, String document)
{
   for each word w in document:
           EmitIntermediate(w, 1);
}
```



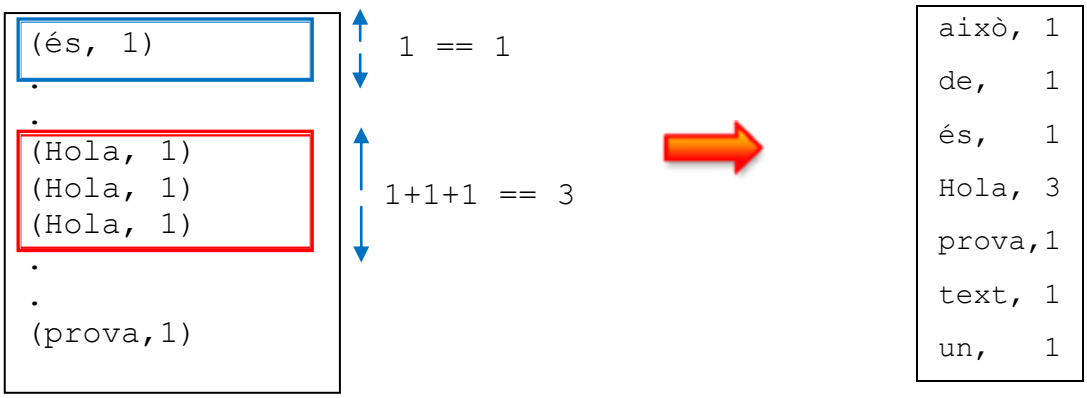
- **Shuffle & Sort** *list(k2,v2) →list(k3,list (vn))*



*Les tuples són agrupades per clau i ordenades per ordre alfabètic.

- **Reduce** *(k3,list(vn))* →*list(vn)*

```
reduce(String word, Iterator partialCounts)
{
  int result = 0;
  for each v in partialCounts:
        result += ParseInt(v);
  Emit(result);
}
```



2.5.3 Framework Hadoop

Hadoop és la implementació més popular del paradigma *MapReduce*. El framework de *Hadoop* és un projecte creat per **Apache** (inspirat en la versió de *Google*) i conté a la mateixa vegada sub-projectes com el **HDFS** (Hadoop Distributed File System) molt similar a l'utilitzat per *Google (*el Google File System *GFS).*

Hadoop va ser creat per **Doug Cutting** (empleat de **Cloudera**) i a l'actualitat existeixen diverses implementacions de codi lliure en tots els *lps* : *C++, Perl, Python, Java etc. Hadoop* és molt útil per projectes *intensius de dades* que requereixen **escalabilitat** i un rendiment eficient. A l'actualitat s'han fet proves amb èxit en clusters superiors als 10.000 nodes (propietat de ***Amazon***).

El framework de *Hadoop* suposa una gran **abstracció per al programador,** el qual ara només s'ha de preocupar per programar les funcions Map i Reduce. Aquesta abstracció és molt potent i facilita tasques tan crítiques com són les **d'assignació i monitorització** de treballs, **distribució de les dades** a través de totes les màquines del cluster utilitzant un **sistema de fitxers distribuït** *(DFS),* un gestor de cues pels treballs etc. A més la plataforma està molt orientada al programador i facilita tota una sèrie d'eines de desenvolupament força intuïtives i útils. Malgrat que existeixen versions de desenvolupament tant per *Windows* com en *Linux*, *Apache* aconsella aquest últim per obtenir un funcionament òptim al cluster.

Una altra avantatge important que dóna el Framework és la **tecnologia Hardware** necessària per fer-lo funcionar. No és necessita un Hardware específic ni últim model ja que funciona perfectament amb tecnologia **x86** amb **disc durs IDE** i una quantitat de **memòria** des dels **512 MB de RAM**.

Figura 2-13. Cluster amb pc's de tecnologia x86

2.5.4 CloudBurst

CloudBurst (el link del seu paper està afegit a l'apartat de referències) és tracta d'una altra aplicació de *Tercera Generació de Read Mapping* que tot i no innovar força en la tècnica de mapeig ni puntuació, si ho fa en la tecnologia i en la forma de programació de l'algoritme.

El seu autor, **Michael C. Schatz**, amb una llarga experiència en el món de la genòmica (Enginyer del **CBCB**, *Center for Bioinformatics & computational biology*), va veure aviat l'aplicació del paral·lelisme en aquest tipus d'algoritme i en especial del paradigma **MapReduce**.

No entrarem en detall del algoritme ja que *CloudBurst* no destaca per la tècnica utilitzada (implementa també l'operació de **seed and extend**) però si comentarem a grans traces la seva aportació a l'Estat de l'Art utilitzant el framework de **Hadoop**.

CloudBurst va saber veure la "**facilitat"** i sobretot la **necessitat** de **paral·lelitzar les tasques** en els algoritmes de Read Mapping i es va programar en base dues classes principals ja conegudes per nosaltres : el **Map** i la classe **Reduce.**

La classe *Map i Reduce* implementen també l'opció de **GAP** i tenen arguments configurables per l'usuari molt interessants des del punt de vista del Enginyer. Arguments molt típics del propi framework de Hadoop com ara:

- Número de Processadors amb els que treballar
- Número de workers disponibles
- Tamany del bloc de disc pel HDFS
- Número de Maps a executar en paral·lel
- Número de Reduces a executar en paral·lel

Si estudiem una mica el *paper* del projecte publicat al abril *del 2009* podem veure gràfiques interessants com la de la escalabilitat:

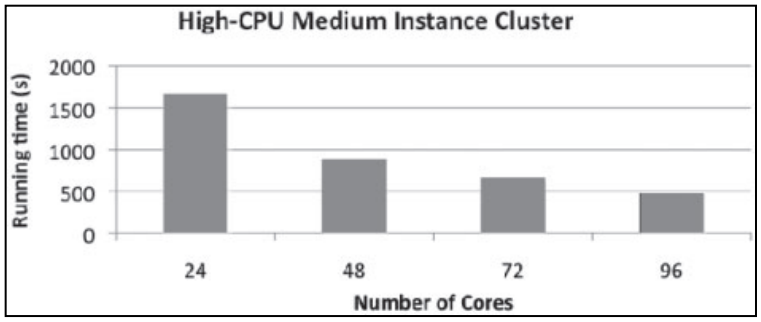


Figura 2-14. Temps d'execució al cluster

A la **figura 2-14** veiem que *CloudBurst* mapejant uns tamanys de fitxer de 7Mb per les mostres de short·Reads i de 49.7 MB del cromosoma 22 pel genoma de referència, escala linealment en temps d'execució amb un índex del 3'5 entre l'execució de 24 i la de 96 cores. És per tant uns bons valors d'escalabilitat.

Per nosaltres *CloudBurst* té un interès doble ja que a més d'aproximar-nos encara més als algoritmes de *Tercera Generació de Seqüenciació,* també ens dóna una bon punt de referència per orientar-nos en la forma de programar del paradigma MapReduce. Per aquest motiu també l'hem escollit com un pilar important per la implementació del nostres algoritme de Read Mapping basat en Hadoop.

Capítol III

Anàlisi i Disseny

Ens trobem ara en un dels punts clau d'aquesta memòria: l'*Anàlisi i el disseny* del *PFC*. L'anàlisi, fortament lligat amb el bloc anterior de l'Estat de l'Art, enumerarà i discutirà la viabilitat de les diferents solucions per a resoldre el problema. El disseny en canvi, tractarà exclusivament i amb detall la **solució proposada**.

3.1 Anàlisi

Al bloc anterior hem pogut veure amb detall gran part de les diferents tècniques de Seqüenciació del genoma Humà. De totes aquestes solucions possibles hi ha algunes més adequades que d'altres.

Per exemple si fem memòria, vam parlar de tot un seguit de solucions molt orientades al camp biològic i químic**.** Aquests tipus de seqüenciació, també anomenades *seqüenciacions químiques*, estableixen unes mancances naturals importants per a nosaltres. Totes elles requereixen d'uns **coneixements científics i biològics** molt importants, que nosaltres com Enginyers informàtics, malauradament no disposem. A més aquest tipus de seqüenciació requereix d'un entorn de treball molt costós i difícil d'aconseguir.

També vam poder veure altres tècniques que resolien el problema de la seqüenciació des d'un punt de vista més genèric i sense tenir un patró de referència. Aquestes **tècniques de novo**, requereixen d'un sistema **Hardware específic molt costós** i es necessitaria d'una formació específica que no tenim. A més, aquests sistemes acostumen a quedar-se ràpidament obsolets ja que van lligats a una metodologia en continu estat de canvi.

Finalment vam estudiar una tecnologia apareguda arrel de l'èxit del **Projecte Genoma Humà.** Aquesta tecnologia, desenvolupada en el que és coneix com la *Tercera Generació de Seqüenciació*, tenia com a base l'Enginyeria informàtica comú. És a dir, un Software que en aquest cas representa els propis algoritmes de Read Mapping i un Hardware Standard que va desde una computadora senzilla d'un sol node fins a un Sistema paral·lel o cluster. Aquesta tecnologia representa una solució molt viable i interessant

per nosaltres per molts motius: la tecnologia de Read Mapping té una precisió molt gran; els algoritmes de Read Mapping es poden modular en fases ben marcades informàticament; treballa amb un entorn **multicluster** que nosaltres tenim etc.

És tracta per tant doncs de la tecnologia més adequada per a les nostres necessitats i la més propera als nostres coneixements d'Enginyers Informàtics. **Però com tota tecnologia, existeixen diferents tècniques o solucions per fer "el mateix".**

Gran part d'aquestes tècniques ja les hem estudiat al capítol de l'Estat de l'Art. Solucions totes elles acceptades però que distaven molt una de l'altra, amb diferències que anaven des de l'entorn d'execució (*MAQ* s'executava en un entorn de **Single Node** i *CloudBurst* en **multicluster**) fins al propi paradigma de programació (*MAQ* escrit en **Perl** i *CloudBurst* amb **Map Reduce**). Tot i així vam veure que malgrat les fortes diferències d'ambdues aplicacions, compartien tècniques i procediments molt similars com ara la tècnica de Seed and Extend així com la **indexació per Taules Hash**.

Evidentment això no significa que *MAQ i CloudBurst* constitueixin totes les solucions existents a la *Tercera Generació de Seqüenciació*. En realitat existeix tota una vessant paral·lela a les aplicacions que utilitzen la indexació per taules Hash. Aquestes són les anomenades aplicacions d'indexació per Arsenal de Sufix o **Suffix Array**. Aquesta branca és relativament recent (finals del 2007) però ja conté algoritmes de prestigi amb resultats realment eficients com **BWA, SOAPv2, Bowtie** etc.

BWA i algoritmes similars treballen amb la tècnica de compressió de dades anomenada **Burrows Wheeler** la qual resulta molt útil per problemes genòmics com els que hem anat veient. Això és degut a que **l'indexació de Burrows Wheeler** ens permet localitzar ràpidament ocurrències de subseqüències dintre d'altres seqüències. És fàcil veure que per al nostre cas entendrem per subseqüència les mostres short·Reads i per seqüència el propi genoma de Referència. A la **figura 3-0** veiem un exemple de la tècnica:

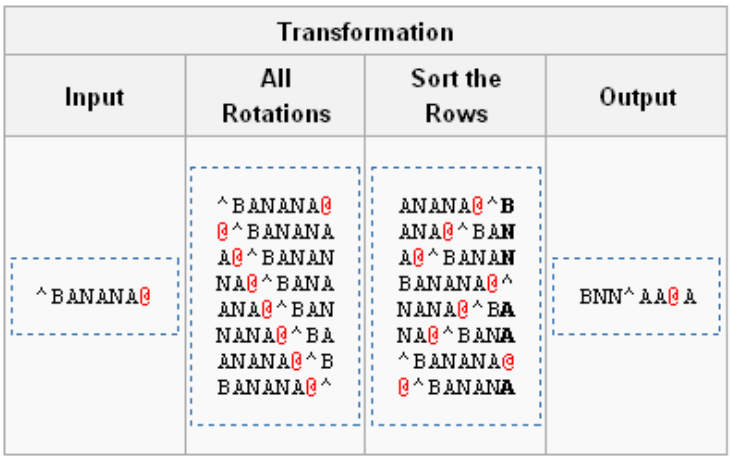


Figura 3-0. Tècnica de compressió Burrows Wheeler. A partir d'una paraula o mostra ens dona totes les possibles representacions per ser localitzada ràpidament en seqüències més llargues que la continguin.

Les tècniques de *Sufix Array* presumeixen de ser **molt eficients en quant a ús de la memòria principal** (es calcula que BWA necessita només 1Gb de RAM per alinear el genoma humà sencer de més de 3 Gbps) i de ser **ràpides** un cop la fase d'indexació prèvia s'ha completat. Però malauradament a l'actualitat encara disposa de molts handicaps seriosos com ara la seva ineficiència per short·Reads de mitja-baixa qualitat (amb molts errors d'extracció) i de la necessitat d'executar d'una fase prèvia d'indexació per a cada experimentació (de lo contrari, sense una fase prèvia d'indexació i amb l'ús de programació dinàmica, la tècnica esdevé molt lenta).

Per aquests i altres motius que es veuran en detall al capítol de Disseny i implementació, **la solució escollida per aquest Projecte serà la del Framework de Hadoop que implementa la tecnologia amb Taules Hash**. Cal tenir present però, que tot i disposar d'un entorn adequat per aquest tipus d'aplicacions, se'ns farà necessari alguna tecnologia de suport que ens ajudi a configurar i solucionar els problemes clàssics. Els entorns paral·lels com sabem, requereixen d'una bona sincronització amb tots els nodes, d'una forta tolerància davant de fallades, d'un gestor de balanceig eficient etc. Per tant ens podria ser molt útil pensar en alguna tecnologia o Framework que ens facilites tota aquesta feina.

3.2 Disseny

Arribem finalment al bloc que tractarà exclusivament i amb detall la **solució proposada** per l'algoritme de *Read Mapping*. Aquest tipus d'aplicacions, com ja s'ha vist al capítol de l'Estat de l'Art, tenen unes característiques molt particulars que les situen en el camp computacional d'**aplicacions intensives de Dades**.

El gran volum de dades d'entrada i l'alt nivell de paral·lelisme trobat en els algoritmes estudiats, fan que pensem en **Hadoop** a l'hora d'escollir un Framework de suport. Aquest Framework s'adapta perfectament a les necessitats generals que anunciàvem a l'apartat d'anàlisi i ens proporciona dues solucions en una: per una banda gestiona tota la part del sistema de fitxers distribuït (**HDFS**) i per l'altra ens ofereix un tipus de programació que explota molt encertadament aquesta tecnologia. En resum:

- El **HDFS** (*Hadoop Distributed File System*)és un sistema de fitxers :
    1. **Distribuit**
    2. Molt **estable**
    3. Segur i tolerant a fallades
    4. Orientat a un **Hardware econòmic**

5. Proporciona una **abstracció gran** (es construeix sobre els discs dels nodes de processament)

- **Hadoop** ens proporciona :

1. Programació functional ben modularitzada : **Map** i **Reduce**
2. Representa el millor de tota la experiència de *Google i Yahoo*
3. **Escalabilitat** per projectes molt grans (*terabytes i petabytes*)
4. Dissenyat per clusters de servidors

Hem escollit l' implementació en Java del Framework degut a ser la versió més actualitzada constantment i la que més difusió té a la comunitat d'usuaris. Ara, al més de Novembre, s'està per la versió estable de **Hadoop 0.19-3** la qual permet llançar treballs desde un sistema de cues convencional (*Hadoop on Demand*) i per aquests motius serà la versió escollida.

Per l'aspecte de l'entorn d'execució dir que es disposa d'un **HPC dedicat** d'un màxim de **8 nodes** al laboratori Q5/0004. Aquest laboratori serà el lloc destinat a realitzar les proves un cop s'hagi programat una primera versió de l'algoritme. Aquest algoritme rebrà el nom de **Map Reduce MAQ,** donat que és tracta d'una versió programada amb el paradigma Map Reduce i amb la tècnica de l'algoritme MAQ com a base.

Hadoop un framework implementat precisament per aquest tipus d'aplicacions resol a la perfecció aquest model d'entorn i resol les problemàtiques clàssiques per aquest entorn.

3.3 Estudi de viabilitat del Projecte

L'apartat del estudi de viabilitat representa un altre bloc important de la memòria ja que és on s'estudia la viabilitat *Real* del Projecte. Per a poder fer-ho correctament, la viabilitat s'aborda des de punts de vista diferents: **tècnic, econòmic, legal i temporal**. Tots ells tracten d'avaluar les avantatges i inconvenients que comporta la realització d'aquest Projecte i per tant determinar les seves possibilitats.

3.3.1 Viabilitat Tècnica

La ***tecnicitat*** és probablement una de les qualitats, entre moltes altres, que millor defineixen a un Enginyer. En el nostre cas com enginyers informàtics, la viabilitat tècnica estarà marcada per dos aspectes importants: la *plataforma utilitzada* i la *tecnologia a emprar*.

- **Plataforma**

Per tal de desenvolupar l'algoritme *Map Reduce MAQ* s'utilitzarà el llenguatge de programació Java i el IDE Eclipse, el qual ofereix un entorn molt adaptat per a **Hadoop**. Per la meva experiència se que Java es tracta d'un *LP* molt intuïtiu i portable. A més s'han desenvolupat nombroses llibreries de Hadoop amb les seves pròpies API's que facilitaran encara més el desenvolupament. Exactament, s'ha decidit treballar amb l'ultima versió estable de la llibrera, la [**Hadoop 0.19.3**].

- **Tecnologia**

En aquest bloc podem distingir dues fases: la **fase de desenvolupament i la fase de proves**. Per a la primera fase dir que és necessitarà un *PC* amb un processador intel x86 amb una memòria *RAM* mínima de 1*Gb* i amb un *S.O unix-like* **Debian** com Ubuntu 9.10. Per a la segona fase, més complexa tecnològicament, és necessitarà un cluster d'uns **4 o 8 nodes** amb tecnologia similar al terminal de desenvolupament.

Pel que fa a les **dades d'entrenament i proves** de l'algoritme, sabem que les dades del genoma i les mostres de ADN estan disponibles a la base de dades del *National Center for Biotechnology Information* (**NCBI**).

Valorant tots els aspectes tècnics i tenint en compte la nostra situació (departament *DACSO*) és pot afirmar que la realització del Projecte és totalment viable. Del contrari, en condicions normals, segurament seria un projecte més costós ja que disposar d'un **cluster** disponible les 24h del dia durant varios mesos pot ser difícil i molt car.

3.3.2 Viabilitat Temporal

Aquesta fase de la viabilitat es va veure amb detall al **capítol I** de la memòria on s'expressava amb detall la duració del Projecte. En hores es va calcular que el Projecte tenia una duració de **612 h** en total i de **65 h** més per a la redacció de la memòria. Per la banda de les hores de supervisió i investigació del director va ser una mica més complicat de calcular però aproximadament vam estimar-ho en un total de **86h**.

Considerant que la nostra disponibilitat és d' aproximadament **4h/dia** durant tot el semestre, el Projecte es considera perfectament assumible.

3.3.3 Viabilitat Legal

La viabilitat legal del Projecte no suposa tampoc un problema. Tots els algoritmes que prenem com a referència han sigut obertament publicats en papers oficials i lliures sota la **llicència GPL**.

En quant a les tècniques i mètodes propis dissenyats i revisats per mi y el meu director Porfidio Hernàndez, també són oberts i es permet la difusió complerta de tots els resultats i codis.

3.3.4 Viabilitat Econòmica

La viabilitat econòmica evidentment també és una part important de la viabilitat final del Projecte. En aquest cas però, resulta bastant senzill el càlcul ja que com hem dit al bloc tècnic, totes les eines de programació utilitzades són gratuïtes, a més del Sistema Operatiu i les dades amb les quals és treballarà.

Per l'aspecte de tecnologia tampoc existirà cap problema ja que el departament *DACSO* disposa de tots els requeriments tecnològics i professionals.

Així doncs la despesa econòmica representarà únicament el cost que suposa tenir a un "*quasi*" Enginyer un número d'hores i a un director que supervisi en tot moment el Projecte. Tenint en compte que el preu actual d'un desenvolupador (sense el títol acabat) oscil·la en *12€/h* i que els honoraris d'un Enginyer amb experiència poden ser de *35€/h* donà uns números de **8124€** per la fase de desenvolupament i memòria i de **3010**€ per les *86h* de supervisió i investigació del director.

És a dir, el Projecte tindria un cost total de **11134€**. Un preu assumible i aprovat pel projecte *Consolider* del Ministeri de Ciència i Tecnologia.

Capítol IV

Implementació

Arribem, ara si, al bloc on tractarem i analitzarem en detall l' implementació del nostre algoritme de **Read Mapping** *MR MAQ*. En primer lloc analitzarem els diferents mòduls principals i les seves funcionalitats a alt nivell. Seguidament farem un repàs de les classes i interfícies Java implementades per dur a terme tota la funcionalitat de l'aplicació. També a més de les classes intervingudes, farem una breu descripció de la interfície de desenvolupament del Framework de **Hadoop** així com de l'entorn d'execució.

4.1 Mòduls Principals

La implementació de l'algoritme és divideix en quatre mòduls diferents:

- **Pre-processament i llançador del treball:** analitza les dades d'entrada i els hi aplica una transformació per facilitar el mapeig. A continuació llança el treball al cluster segons la configuració del mòdul de Sintonització.
- **Sintonització del Framework:** el Framework de Hadoop té un total de **123** paràmetres configurables per part de l'administrador i l'usuari.

- **Classes de Suport al mòdul Map Reduce:** són classes que defineixen el tipus de dades , o **stream**, de la comunicació entre el Map i el Reduce. Cada classe a més té una sèrie de funcions que facilita el treball amb els altres mòduls.
- **Execució Map Reduce:** és tracta del mòdul principal de l'algoritme que implementa les fases de **Map, Shuffle &Sort i Reduce**.

4.1.1 Mòdul Pre-processament i llançador del treball

Aquest mòdul programat a la classe **`Mrmaq.java`** té dues funcionalitats importants:

1. Analitza les dades d'entrada, és a dir, les mostres **short·Reads i el Genoma de referència.** Per les mostres curtes l'usuari pot escollir el número de **bps** a analitzar segons el paràmetre *ReadLength*. Aquest paràmetre pot ser molt útil per biòlegs experts ja que està demostrat que per les últimes bases de les mostres s'acostuma a cometre un índex d'error d'extracció superior a la part inicial. D'aquesta manera si considerem un fitxer d'entrada amb unes mostres de longitud de 36 bps, si l'usuari escull l'opció de 30 bps aleshores s'estarien descartant automàticament les últimes sis bases de cada mostra. Per altra banda, el mòdul s'encarrega també de preparar el genoma per a facilitar el seu mapeig amb les mostres. Per tal de fer-ho es procedeix a subdividir-lo en *n* fragments de longitud també *ReadLength* , on *n* és la ***Longitud del tamany total del fitxer / Mod [Readlength].*** Podem veure un exemple d'aquesta operació en les figures següents:

```
>Chr_22_14-09-2001 Release 3, [nucleotoides 13100001-47848585]

GATCTGATAAGTCCCAGGACTTCAGAAGAGCTGTGAGACCTTGGCCAAGTCACTTCCTCC
TTCAGGAACATTGCAGTGGGCCTAAGTGCCTCCTCTCGGGACTGGTATGGGGACGGTCAT
GCAATCTGGACAACATTCACCTTTAAAAGTTTATTGATCTTTTGTGACATGCACGTGGGT
TCCCAGTAGCAAGAAACTAAAGGGTCGCAGGCCGGTTTCTGCTAATTTCTTTAATTCCAA
GACAGTCTCAAATATTTTCTTATTAACTTCCTGGAGGGAGGCTTATCATTCTCTCTTTTG
GATGATTCTAAGTACCAGCTAAAATACAGCTATCATTCATTTTCCTTGATTTGGGAGCCT
AATTTCTTTAATTTAGTATGCAAGAAAACCAATTTGGAAATATCAACTGTTTTGGAAACC
TTAGACCTAGGTCATCCTTAGTAAGATCTTCCCATTTATATAAATACTTGCAAGTAGTAG
TGCCATAATTACCAAACATAAAGCCAACTGAGATGCCCAAAGGGGGCCACTCTCCTTGCT
TTTCCTCCTTTTTAGAGGATTTATTTCCCATTTTTCTTAAAAAGGAAGAACAAACTGTGC
CCTAGGGTTTACTGTGTCAGAACAGAGTGTGCCGATTGTGGTCAGGACTCCATAGCATTT
CACCATTGAGTTATTTCCGCCCCCTTACGTGTCTCTCTTCAGCGGTCTATTATCTCCAAG
AGGGCATAAAACACTGAGTAAACAGCTCTTTTATATGTGTTTCCTGGATGAGCCTTCTTT
TAATTAATTTTGTTAAGGGATTTCCTCTAGGGCCACTGCACGTCATGGGGAGTCACCCCC
AGACACTCCCAATTGGCCCCTTGTCACCCAGGGGCACATTTCAGCTATTTGTAAAACCTG
```

Figura 4-1. Genoma de Referència original

```
>Chr_22_14-09-2001 Release 3, [nucleotoides 13100001-47848585]

*1]  GATCTGATAAGTCCCAGGACTTCAGAAGAGCTGTGA
*2]  GACCTTGGCCAAGTCACTTCCTCCTTCAGGAACATT
*3]  GCAGTGGGCCTAAGTGCCTCCTCTCGGGACTGGTAT
*4]  GGGGACGGTCATGCAATCTGGACAACATTCACCTTT
*5]  AAAAGTTTATTGATCTTTTGTGACATGCACGTGGGT
*6]  TCCCAGTAGCAAGAAACTAAAGGGTCGCAGGCCGGT
*7]  TTCTGCTAATTTCTTTAATTCCAAGACAGTCTCAAA
*8]  TATTTTCTTATTAACTTCCTGGAGGGAGGCTTATCA
*9]  TTCTCTCTTTTGGATGATTCTAAGTACCAGCTAAAA
*10] TACAGCTATCATTCATTTTCCTTGATTTGGGAGCCT
*11] AATTTCTTTAATTTAGTATGCAAGAAAACCAATTTG
*12] GAAATATCAACTGTTTTGGAAACCTTAGACCTAGGT
*13] CATCCTTAGTAAGATCTTCCCATTTATATAAATACT
*14] TGCAAGTAGTAGTGCCATAATTACCAAACATAAAGC
*15] CAACTGAGATGCCCAAAGGGGGCCACTCTCCTTGCT
*16] TTTCCTCCTTTTTAGAGGATTTATTTCCCATTTTTC
*17] TTAAAAAGGAAGAACAAACTGTGCCCTAGGGTTTAC
*18] TGTGTCAGAACAGAGTGTGCCGATTGTGGTCAGGAC
*19] TCCATAGCATTTCACCATTGAGTTATTTCCGCCCCC
*20] TTACGTGTCTCTCTTCAGCGGTCTATTATCTCCAAG
*21] AGGGCATAAAACACTGAGTAAACAGCTCTTTTATAT
*22] GTGTTTCCTGGATGAGCCTTCTTTTAATTAATTTTG
```

Figura 4-2. Genoma després d'aplicar el Modul de Pre-processament

2. La classe **`MRmaq.java`** també s'encarrega del llançament del treball, o **job**, al mòdul principal Map Reduce. Per gestionar-ho es recolza amb el segon Mòdul de sintonització on s'especifiquen tots els paràmetres necessaris per al Framework de Hadoop el qual treballa amb el HPC.

4.1.2 Mòdul Sintonització del Framework

El mòdul de sintonització **HadoopEclipse.0-19.jar** és troba ja implementat pel propi Framework de Hadoop. Aquest mòdul resulta molt útil i potent ja que disposa de més de 123 paràmetres de configuració per part de l'administrador. A més, el treball amb aquest mòdul resulta senzill ja que s'incrusta a la mateixa plataforma de desenvolupament com un plugin d'Eclipse més. Amb aquest plugin també se'ns facilita la gestió de fitxers del **HDFS** permetent-nos així la pujada/baixada dels arxius que vulguem afegir al projecte i a l'entorn distribuït.

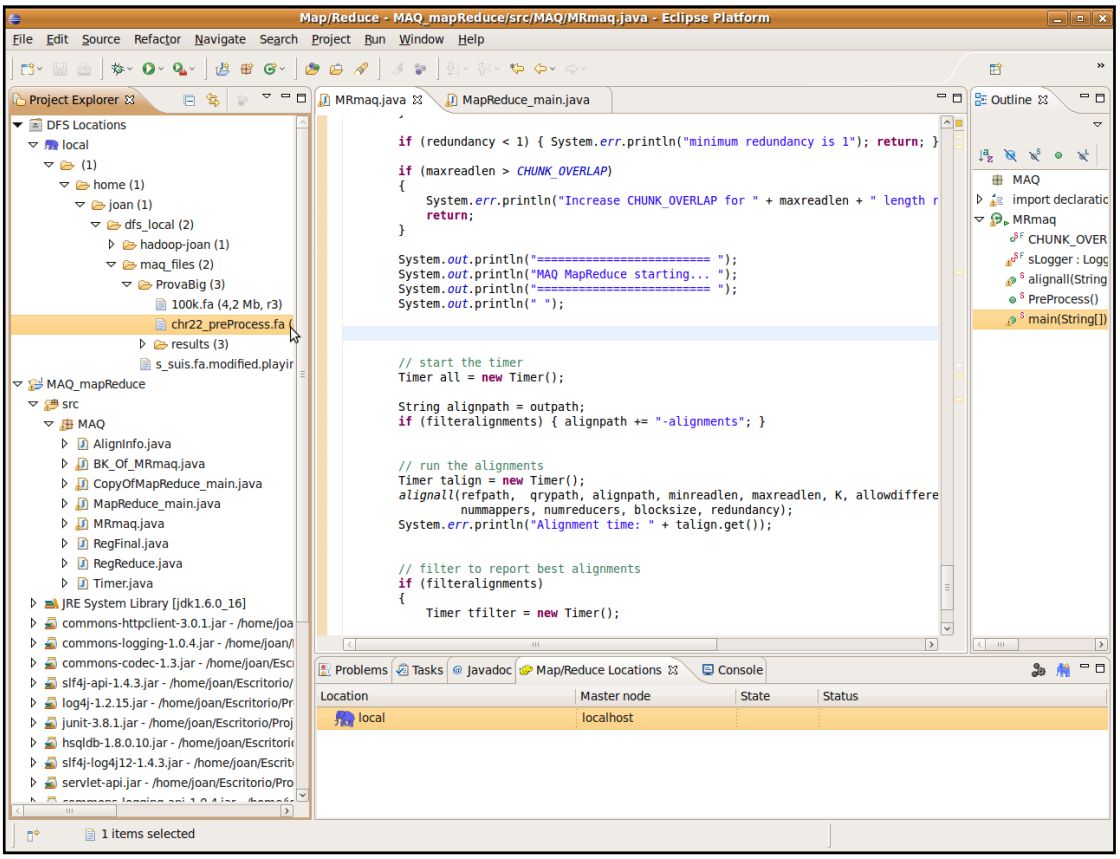


Figura 4-3. Entorn de desenvolupament de Hadoop

A la **figura 4-3** podem veure com amb aquest plugin podem controlar tots els mòduls de l'aplicació d'una forma ràpida i unificada. A la part inferior esquerra del **explorador de Projectes** veiem l'arbre jeràrquic del codi font de l'algoritme, idèntic al d'un projecte Java habitual. A la part superior d'aquesta mateixa vista trobem l'arbre jeràrquic dels fitxers que es troben al **HDFS**. Podem veure com en aquests moments tenim disponibles els fitxers del cromosoma 22, ja pre-processat, com a genoma de referència i les mostres short·Reads a l'arxiu 100k.fa (**100.000** mostres en format Fasta).

A la part central inferior veiem la vista relacionada amb el **HPC de Hadoop**. Aquest mòdul, evidentment relacionat amb el **HDFS** anteriorment mencionat, guarda tots els paràmetres de configuració del cluster amb l'execució de l'aplicació de Read Mapping.

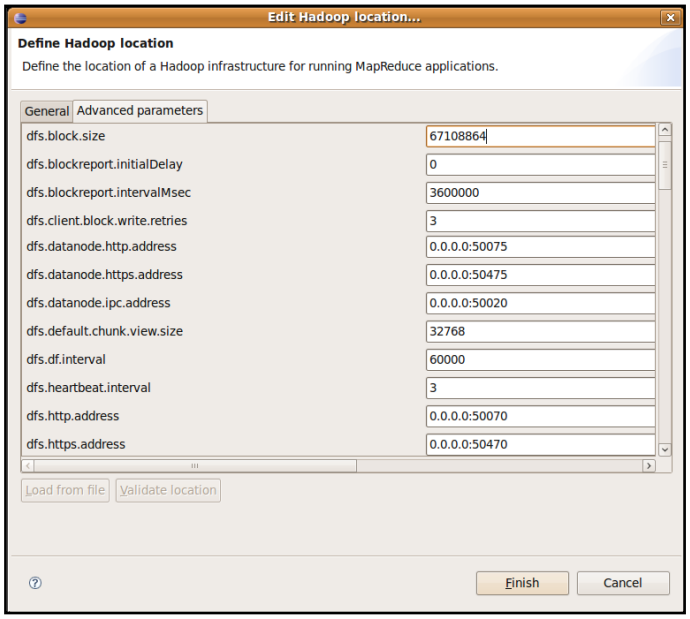


Figura 4-3.2. El Modul Sintonització té més de 123 paràmetres configurables

4.1.3 Mòdul classes de Suport

La programació amb el paradigma Map Reduce requereix d'un estil determinat i especificat per la pròpia **API [Hadoop 0.19.2].** Hadoop aconsella utilitzar tota la potència del *Orientat a Objectes* per poder extreure el màxim rendiment del Framework i del propi cluster. Nosaltres conscients d'aquest bon ús hem orientat l'algoritme totalment a aquest paradigma i hem programat diverses classes per tal de modularitzar el codi el màxim possible.

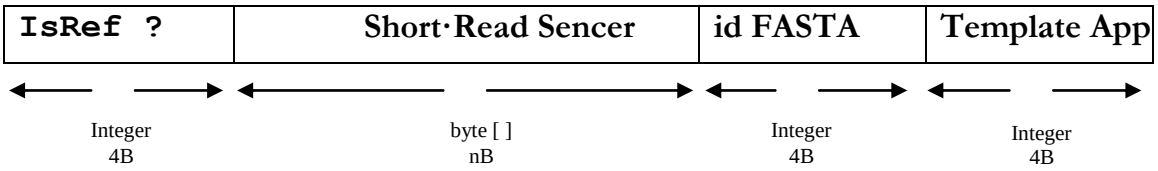
Una de les especificacions del Framework és la comunicació entre les funcions Map i Reduce. Com ja hem pogut veure en blocs anteriors, aquesta comunicació es basa principalment en tuples *(clau,valor).* **Però què passa si en el camp *v* volem afegir més dades?** El nostre problema és molt complexe i per tant se'ns farà necessari passar altres camps al reduce. Nosaltres en realitat el que volem passar són **tuples de tuples** de l'estil *(c, (v_1,v_2,...,v_n))* o fins i tot **tuples, de tuples de tuples** *(c, (v_1,v_2(v_1_1,v_1_2,v_1_n))).*

Però aquest format, més aviat conceptual, pot ser perillós si no es controla adequadament el tipus, la longitud o els continguts de cada camp. En aquest cas ens podríem trobar amb problemes desastrosos a la fase de Reduce i molt difícilment detectables (S'envien milions de tuples en la comunicació entre el Map i el Reduce). És per aquest motiu que **Hadoop** aconsella un mètode de *bona*

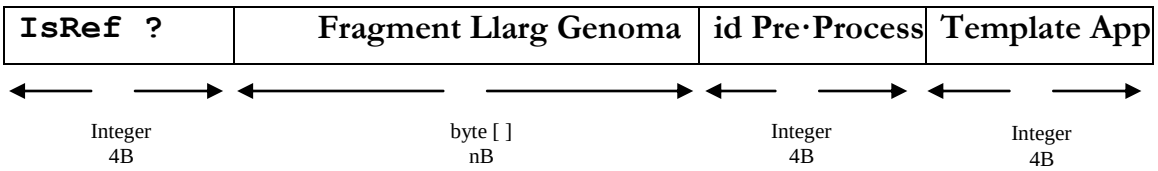
praxis per aquesta comunicació. Hadoop aconsella definir un fluxe o **stream de dades** controlat i ben estructurat per a la comunicació entre Map i Reduce. Aquest fluxe a més existeix la possibilitat de poder comprimir-ho en formats per tal d'agilitzar el seu enviament per la xarxa.

En el nostre cas hem definit dos tipus de fluxes o streams per a la sortida del Map. Un per cada registre de cada fitxer d'entrada, és a dir, un stream per a cada registre del fitxer del genoma de referència i altre pel fitxer de les mostres short·Reads. Aquests streams fan referència a les classes **`RegReduce.java`** i **`RegFinal.java`** del projecte i tenen l'estructura següent:

- Registres pel fitxer de mostres curtes:



- Registres pel fitxer del genoma de referència pre-processat:



Dintre d'aquestes classes existeixen també funcions que faciliten operacions produïdes al mòdul principal d'execució Map Reduce. Una de les més importants és la funció comptador de missmatches **missmatchesCount,** la qual calcula el número de missmatches entre la llavor, en aquest cas la *key* de la tupla, i el fragment llarg del camp de *value*. Aquesta funció calcula l'extensió de la llavor per així determinar la qualitat del alineament.

4.1.4 Mòdul d'Execució Map Reduce

La classe **`MapReduce_main.java`** és la classe que conté el codi principal de l'aplicació. Aquesta classe incorpora alhora les classes **`MapClass i ReduceClass`** que hereden de la classe **`MapReduceBase i`** implementen les interficies **`Mapper i Reducer`** respectivament.

Com ja vam explicar a la descripció de l'Estat de l'Art del Framework, tot treball llançat sota el paradigma Map Reduce consta de quatre fases importants: **Map, Shuffle, Sort i Reduce**. De les quatre fases el programador només s'ha de preocupar d'implementar el Map i el Reduce, les altres dues ja són implementades automàticament pel propi Framework de Hadoop.

- **Map**

  La fase Map de l'algoritme és implementada per la classe **`MapClass`** del projecte. Aquesta classe com hem dit hereda directament de la classe *MapReduceBase* i implementa la interfície *Mapper*. El treball amb aquestes classes fa que tot el codi sigui organitzat i **fortament tipat**. La classe MapClass consta de tres arguments principals: la **clau** de tipus *LongWritable*, el **valor** de tipus *Text* i el **OutputCollector** el qual escriura els resultats intermedis del map i que més tard rebrà el Reduce.

  La classe MapClass consta d'una funció principal **map** la qual executa unes tasques molt determinades. Aquestes operacions variaran respecte el fitxer d'entrada que s'estigui tractant en aquell moment i les podem resumir a alt nivell en dos operacions diferents:

  1. Quan la lectura correspongui al fitxer de les mostres short·Reads l'algoritme agafarà un a un tots els registres del fitxer i construirà tuples en forma de *(clau,valor)*. Aquestes tuples es composaran per la *clau,* calculada per una funció d'**aplicació de templates** semblant a la de **MAQ**, i el *valor* el qual estarà definit per l'estructura marcada al **mòdul de classes de Suport**. D'aquesta manera i com vam veure amb MAQ, segons el numero de **Gaps** seleccionats per l'usuari s'aplicaran un número o altre de templates, el que es traduirà per a nosaltres en un numero major o menor de tuples generades.

  2. Si la lectura correspòn a registres del genoma de referència, aleshores el tractament de la informació varia sensiblement. En aquest cas sabem que els registres disposen d'un *identificador "local"* seguits per un fragment d'ADN que correspòn just a la posició anterior del següent registre del fitxer. Per aquests registres es procedirà a aplicar també la funció d'aplicació de templates per la *clau* i a afegir el valor dels identificadors locals al value.

- **Reduce**

La fase Reduce és implementada per la classe **`ReduceClass`** del projecte i també hereda de la classe *MapReduceBase.* La classe *ReduceClass* implementa la interfície *Reducer* i té com a paràmetres principals, al igual que el Map, la **clau**, el **valor** i el **OutputCollector** que guardarà els resultats definitius de l'algoritme.

La fase Reduce és situa després de la de **Shuffle i Sort**, de manera que el conjunt d'entrada de la classe ReduceClass estarà format per grups de registres on la clau coincideix. **Aquest concepte és fonamental per entendre l'algoritme i les avantatges que proporciona el Framework de Hadoop**, ja que la propia fase Shuffle ens agruparà tot el conjunt generat pel map en grups, on cada un representarà per si mateix coincidencies de les mostres Short·Reads amb el genoma de referència. Per aquest motiu és important especificar també el tipatge del paràmetre *valor, el* qual en aquest cas, a diferència del Map, serà de tipus **Iterator<Valor>**. D'aquesta manera ens serà més fàcil d'iterar mitjançant un bucle tots els registres entrants del grup a la classe i aplicar les diferents funcions.

Un cops arribats a aquest punt és fàcil veure com el fluxe restant de la classe Reduce es limitarà a identificar grups composats per mostres Short·Reads i fragments del genoma a la vegada. De lo contrari, les altres coincidències o grups no tenen massa sentit per nosaltres ja que el nostre objectiu es calcular l'alineament de les mostres sobre el genoma de referència. Aquest alineament serà exactament el que s'escriurà al arxiu resultat final i tindrà l'estructura següent:

Fragment i del Genoma :

- **Short Read j** : sentit de lectura, num de Missmatches , puntuació
- **Short Read z**: sentit de lectura, num de Missmatches , puntuació

.

.

Fragment i +1 del Genoma :

- **Short Read w** : sentit de lectura, num de Missmatches , puntuació

.

.

4.2 Diagrama de classes

A la **figura 4-4** podem veure el diagrama de classes del projecte *Map Reduce MAQ*. Amb el diagrama podem apreciar totes les relacions que s'han anat comentant a cada mòdul i les funcions més significatives de cada un.

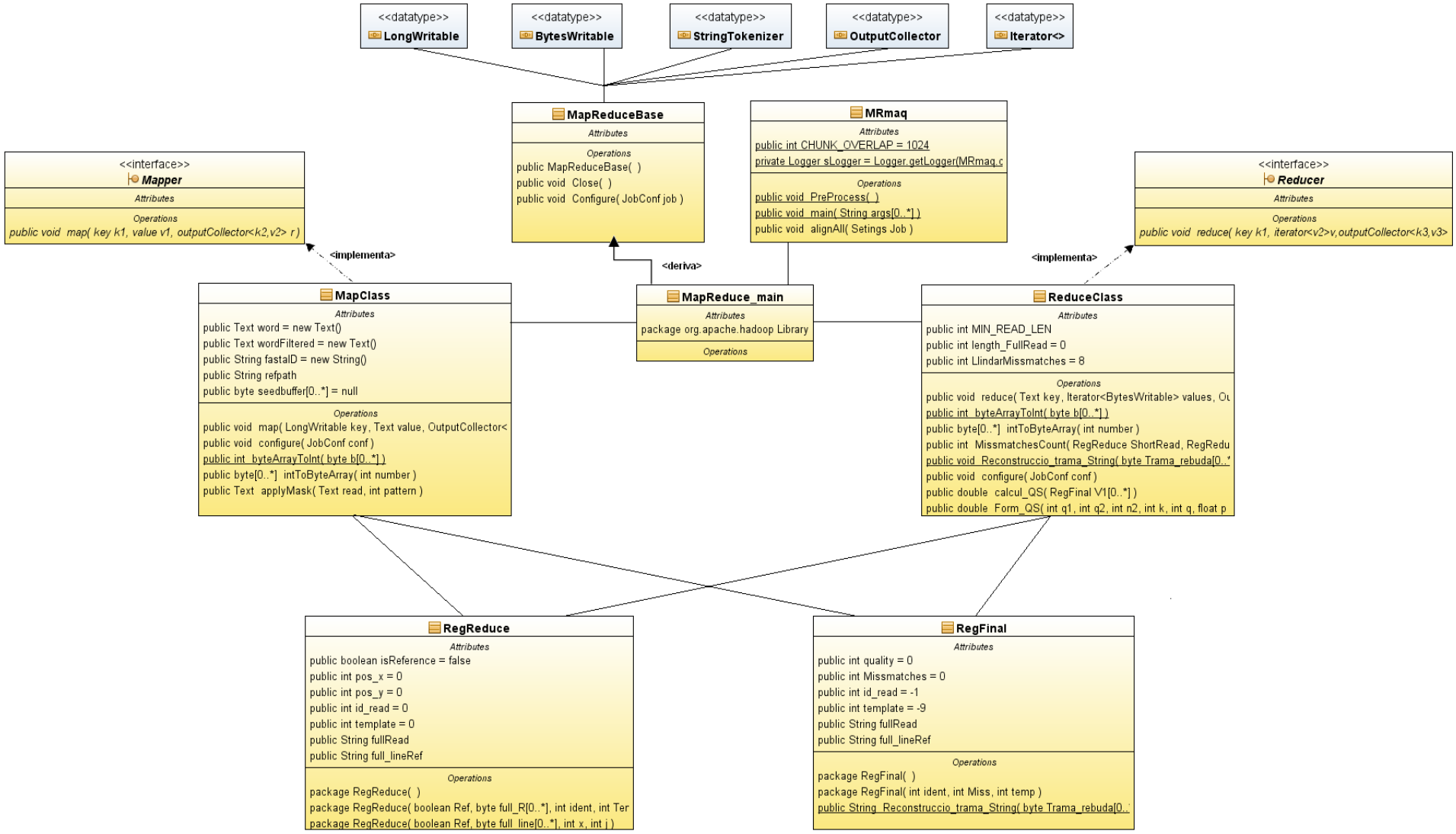


Figura 4-4. Diagrama de classes del Projecte MR Maq

Capítol V

Experimentació Realitzada i Resultats obtinguts

Resulta essencial realitzar una fase exhaustiva d'experimentació per poder determinar el correcte funcionament del projecte. Aquestes proves mesuraran l'aplicació d'una forma fiable i formal desde diferents punts de vista. Serà per tant aquest, un capítol important ple de resultats i números que ajudaran a comprovar el bon assoliment, o no, dels objectius inicials marcats. Per tal d'organitzar totes les proves a les que ha sigut sotmès l'algoritme les hem agrupat en dos grans blocs; el primer assegurarà un correcte funcionament bàsic de l'algoritme i el segon avaluarà l'assoliment d'objectius més complexos com l'eficiència, l'escalabilitat i el paral·lelisme en un entorn **Paral·lel real.**

5.1 Definició de l'entorn

És important definir detalladament l'entorn d'execució on es realitzarà l'experimentació ja que aquest entorn determinarà en gran mesura el valors finals dels resultats.

En el nostre cas, al tractar-se d' una aplicació complexa amb molts i diferents objectius, hem definit dos tipus diferents d'entorn per dos tipus d'experimentacions. El primer estarà destinat a les proves del primer bloc on l'algoritme serà avaluat bàsicament per a determinar el seu bon funcionament lògic en un únic node. Aquestes proves resultaran essencials per a poder passar a un **entorn multicluster** on la complexitat de l'algoritme pujarà notablement i els objectius seran més difícils d'assolir.

▪ Entorn de node únic (Single Node)

Les proves realitzades en aquest entorn ens donaran la informació valuosa de saber si l'algoritme funciona correctament a nivell lògic en un entorn controlat d'un únic node. Aquest entorn és evidentment molt més senzill a nivell de configuració i sintonització que l'entorn multicluster i estarà composat per un node amb les següents característiques:

-**Processador Pentium D** de **doble nucli** de **3,14 Ghz** cada un.

-Memòria **RAM de 1,00 Gb** on 256 Mb seran dedicades exclusivament a la màquina virtual de Java (JVM) necessària per Hadoop.

-**2048 Kb de memòria cau** en dos nivells (**L1** i **L2**) per core.

- Disc dur **SATA de 80 Gb**.

-O.S. **Debian Ubuntu 9,04** amb el kernel **2.6.28-11.37.**

-**Hadoop** versió estable **0.19.2** compilada al juliol del 2009.

-**Jdk 1.6** actualització 17.

▪ Entorn de node múltiple (Multicluster)

El departament *DACSO* disposa de diversos clústers dedicats per als seus estudiants de màster i doctorat. Un d'aquests, el del laboratori Q5/0004, ens ha sigut facilitat durant la llargada de tot el semestre per poder realitzar totes les proves necessàries. Aquest clúster es composa de 4 nodes idèntics amb les especificacions següents:

-**Processador Pentium D** de **doble nucli** de **3,40 Ghz** cada un.

-Memòria **RAM de 1,00 Gb** on 256 Mb seran dedicades exclusivament a la màquina virtual de Java (JVM) necessària per Hadoop.

-**2048 Kb de memòria cau** en dos nivells (**L1** i **L2**) per core.

- Disc dur **IDE** amb una partició física de **15 Gb**.

-O.S. **Debian Ubuntu 9,04** amb el kernel **2.6.28-11.37.**

-**Hadoop** versió estable **0.19.2** compilada al juliol del 2009.

-**Jdk 1.6** actualització 17.

A la **figura 5-1** veiem la imatge real del HPC on s'executaran les proves per un entorn multicluster.

Figura 5-1. HPC del lab Q5/0004

5.2 Bloc 1: Proves de funcionament Bàsic

5.2.1 Genoma curt i conjunt de mostres reduït

El que es pretén amb aquesta primera prova és mesurar la funcionalitat bàsica de l'algoritme per a un conjunt petit de mostres short·Reads i per a un genoma de referència controlat. Com sabem i hem vist en detall en capítols anteriors, l'algoritme MR MAQ calcularà (o hauria de calcular) l'alineament més òptim per totes les mostres sobre el genoma de referència. És a dir, l'algoritme ha de calcular la posició i la puntuació de mapeig per totes aquelles mostres que directament **mapejen** el genoma de referència.

Per tal de fer més senzill l'anàlisi de resultats hem preparat una experimentació amb unes dades d'entrada teòriques **100% mapejables** (a la pràctica això no es dóna mai). És a dir, hem preparat un conjunt de mostres short·Reads on cada una sense excepció mapeja el genoma de referència. Resulta molt fàcil adonar-se que aquestes mostres poden ser "construides" doncs literalment amb el propi genoma de referència. D'aquesta manera a més d'assegurar-nos que mapejaran el genoma de referència ens assegurem que l'alineament serà òptim ja que la correspondència serà exacta.

A nivell d'esquema l'experimentació queda com el de la **figura 5-2**:

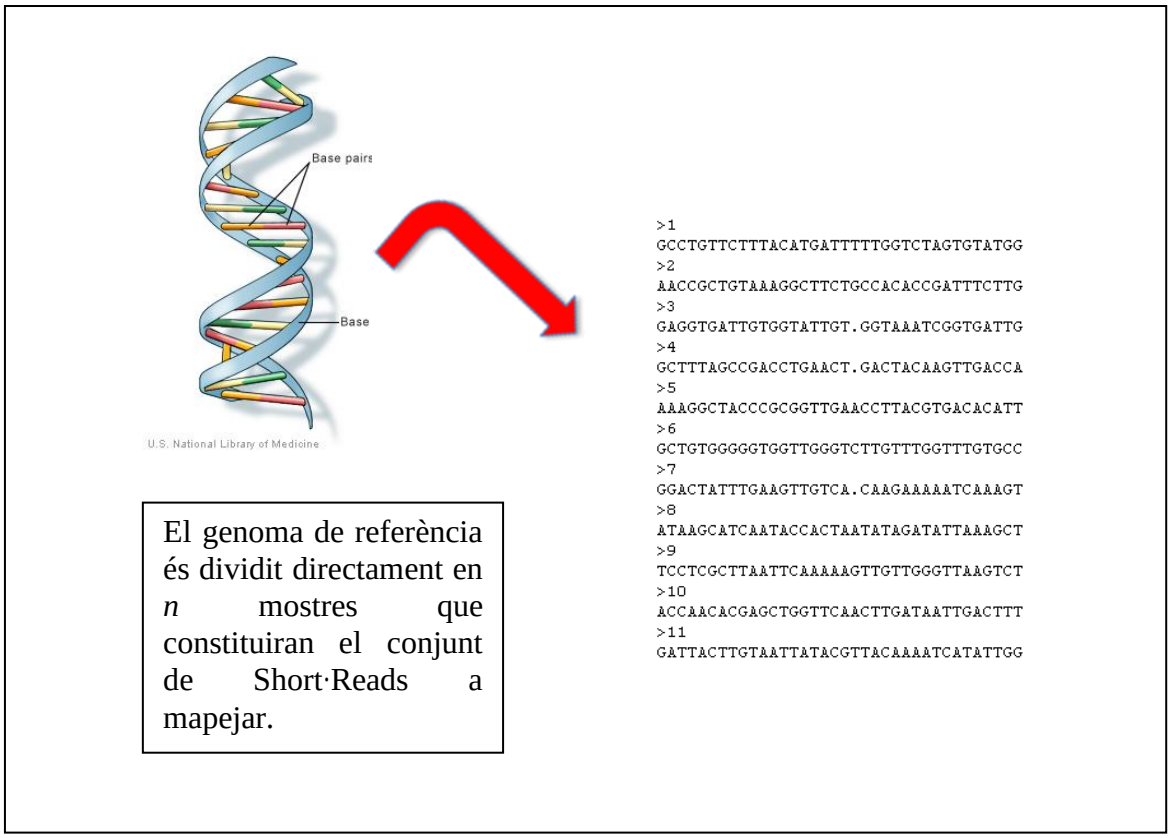


Figura 5-2. Experimentació 1 del Bloc 1

A la experimentació es treballarà amb un fragment del genoma del estreptococ del grup A (**Streptococcus pyogenes**). Aquest genoma identifica a les bactèries comunes de l'àcid làctic i té una longitud total de **4 x10^6 bps**, un tamany molt petit comparat amb el del genoma humà (recordem que estava al voltant de les **3Gbps**). Tot i la seva petita dimensió nosaltres treballarem amb tant sols un fragment de **1080 bps** del qual construirem **30 mostres** short·Reads de **36bps de longitud** cada una.

Per aquesta primera prova els valors de **sintonització del Framework** de Hadoop no seran importants ja que encara no estem mesurant el rendiment de l'aplicació. Per aquest motiu només es setejaran dos paràmetres : el **número de Maps** i el **número de Reduces**, els quals es setejaran **a un** per ambdós paràmetres.

A les **figures 5-3 i 5-4** podem veure l'evolució de l'experimentació:

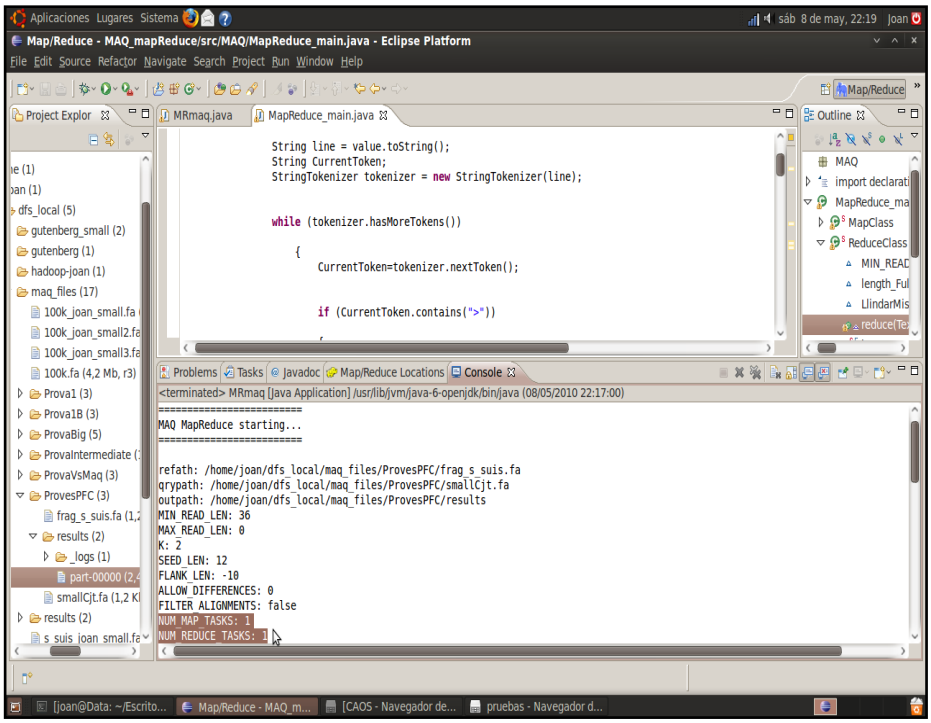


Figura 5-3. Inicialització Prova 1

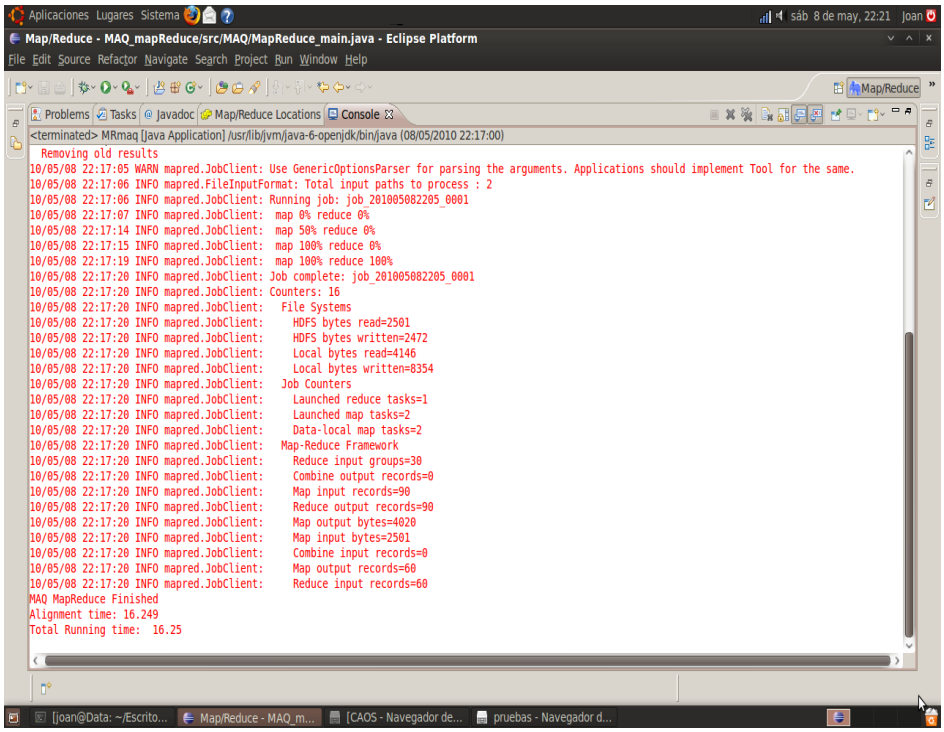


Figura 5-4. Execució Prova 1

A la **figura 5-3** podem apreciar el nom dels arxius amb els que s'està treballant: ***frag_s_suis.fa*** el fragment curt del estreptococ com a genoma de referència i ***smallCjt.fa*** les 30 mostres short·Reads. També s'aprecia el número de maps i de reduces (**NUM_MAP_TASKS** i **NUM_REDUCE_TASKS**).

A la **figura 5-4** observem l'evolució del job, primer per a la fase Map i a continuació, un cop finalitzada la fase de Map, per la fase Reduce. Podem apreciar com el job finalitza en tot just 16 segons, un temps que de moment, com hem dit anteriorment no valorarem gaire ja que això serà qüestió de les següents proves del capítol.

L'execució del job dona com a resultat l'arxiu *part-00000* amb els següents valors:

```
joan@data:\$HOME> cat part-000000

Genoma : 1
    .-> Short Read : 1 , + ,0 , 100.0

Genoma : 2
    .-> Short Read : 2 , + ,0 , 100.0

Genoma : 3
    .-> Short Read : 4 , + ,0 , 100.0

Genoma : 4
    .-> Short Read : 4 , + ,0 , 100.0


Genoma : 5
    .-> Short Read : 5 , + ,0 , 100.0
.
.
.
Genoma : 30
    .-> Short Read : 30 , + ,0 , 100.0
```

Podem observar com els resultats són els esperats; **cada short·Read mapeja** en una única posició del genoma de referència (en experimentacions reals el mapeig pot ser múltiple) on aquesta posició, per aquest cas, és seqüencial. A més, també podem veure com el **'+'** indica que les seqüències s'han analitzat en sentit reverse, és a dir, **d'esquerra a dreta**, i que cada mapeig té **0 missmatches** a la funció d'extensió i per tant té una puntuació d'alineament de 100, és a dir, un **alineament perfecte**.

Aquesta experimentació demostra que la funcionalitat lògica bàsica de l'algoritme funciona correctament i això ens dona la possibilitat de continuar amb més experimentacions que mesuraran altres fases de l'algoritme.

5.2.2 Execució i comparació amb MAQ

MAQ és un algoritme programat amb un paradigma molt diferent al de *Map Reduce*. Recordem que aquest algoritme estava implementat per un **entorn d'execució** d'un **node únic**. A més la implementació està feta per *lps* diferents al nostre (Java) com ara: *c, c++ i Perl*. Tot i així malgrat les fortes diferencies algorítmiques, es tracta d'un algoritme de *Read Mapping* d'alineament global, motiu pel qual seria lògic esperar un resultat, sinó idèntic, molt similar al de la nostra aplicació

Per aquest i altres motius (**MAQ** és l'algoritme de referència en la comunitat genòmica i al **MR MAQ** hi ha funcions que s'han basat fortament en el seu estil) hem trobat interessant afegir unes proves d'execució que contrastin aquests resultats. Alhora aquesta experimentació ens servirà per reforçar el correcte funcionament lògic de la nostra aplicació.

MAQ, al igual que la nostra aplicació, treballa amb dos fitxers d'entrada: el genoma de referència (format ***FASTA***) i les mostres short·Reads (format ***FASTQ***). Aquest últim format és molt similar al *FASTA* però inclou paràmetres específics per a la etapa de Score. Aquesta funcionalitat és recent dels seqüenciadors més moderns i permet calcular el % de fiabilitat amb el cada base ha sigut extreta de les mostres. Per aquest motiu, les mostres short·Reads en format *FASTQ* tenen una línia més per cada registre on s'indica la qualitat de cada base.

```
@identificador length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+identificador length=36 (opcional)
IIIIII4I9IIIIIIIIIIIIIIIIIII8I49IC
```

Figura 5-5. Mostra FASTQ

A la **figura 5-5** veiem com l'aparença dels short·Reads en format *FASTAQ* és molt similar als que ja coneixíem però inclouen alguns paràmetres informatius nous com el de la longitud de les mostres (útil per seqüenciadors que poden prendre simultàniament mostres de diferents tamanys) i la qualitat a la quarta línia (de la mateixa longitud que la primera). **'I'** assegura que la lectura de la base no comet errors i valors ≠ **'I'** indica el % de possibilitats d'error en la lectura.

Aquesta diferència és salva per a la experimentació ja que existeixen nombrosos convertidors de format ***FASTA->FASTQ*** i ***FASTQ->FASTA***. El que molt probablement si afectarà és en la etapa de score, la qual és molt particular per cada aplicació. Per aquest motiu farem una comparació de resultats d'ambdues aplicacions tenint en compte només la fase de Mapeig (la més important) i descartarem les variacions en el score.

Per aquesta prova hem treballat amb la **versió 0.7.1** de *MAQ* escrit en *Perl* , el genoma del estreptococ del grup A (**Streptococcus pyogenes**) sencer i un conjunt reduït de mostres preparat per aquesta experimentació. Aquestes mostres, en format *FASTA* per a la nostra aplicació i *FASTQ* per *MAQ*, tenen la particularitat de mapejar el genoma (amb diferents graus de coincidència o de no mapejar en absolut. És a dir, s'han preparat 50 mostres short·Reads on 35 mapejen el genoma del estreptococ i 15 que no ho fan.

La forma d'executar la prova amb l'entorn Map Reduce és idèntica a la de la primera experimentació. Per aquest cas també s'ha decidit treballar amb un únic node i amb la mateixa configuració dels paràmetres d'usuari **NUM_MAP_TASKS** i **NUM_REDUCE_TASKS** (ambdós setejats a un).

Per la banda de *MAQ* els passos que s'han seguit són els següents:

1. Carreguem el genoma de referència al format intern de MAQ.

```
> gzip -dc Streptococcus.fa.gz | maq fasta2bfa - ref.bfa
```

-- 1 Genome sequence was loaded.

2. Carreguem les mostres short·Reads en format FASTQ al format intern de MAQ.

```
> gzip -dc SmallCjt_4rMAQ.filtered.fq.gz | maq fastq2bfq -
SmallCjt_4rMAQ.filtered.bfq
```

-- finish writing 'SmallCjt_4rMAQ.filtered.bfq'
-- 50 sequences were loaded.

3. Executem l'algoritme indicant el **número de Gaps** (3), la **longitud** desitjada dels **short·Reads** (sencera, és a dir, **36 bps**), els **fitxers de sortida** on s'imprimiran els resultats i evidentment el nom del fitxer del genoma de referència i el de les mostres.

```
>maq map -n 3 -e 36 -u experiment.a.unmap experiment.a.map
 ref.bfa SmallCjt_4rMAQ.filtered.bfq
```

-- maq-0.7.1 [ma_load_reads] loading reads...
32,47 sec
Alignment Successful
...

Abans de comparar les sortides d'ambdues aplicacions cal tenir en compte que MAQ fa una lleugera diferència a l'hora de tractar els resultats. MAQ classifica totes les mostres short·Reads d'entrada en dos grans grups: els que **SI** mapejen el genoma i els

que **NO** ho fan. La nostra aplicació en canvi, com hem vist anteriorment, només imprimeix aquells resultats que mapejen el genoma de referència (cas de **HIT**).

Tot i així és fàcil veure que serà una característica poc important a l'hora de contrarestar resultats.

Figura 5-6. Classificació MAQ i classificació MR MAQ

L'aplicació *MAQ* disposa d'eines molt interessants com l'aplicació ***MAQView*** *(figura 5-7). MAQView* és una aplicació que utilitza **OpenGL** per donar-nos una interfaç gràfica del mapeig. Amb ella podem veure ràpidament el nostre genoma de referència amb les seqüències que han sigut mapejades, el seu grau de solapament, la qualitat de les seqüències, les posicions d'errors etc. Malauradament MR MAQ no disposa de suport gràfic però podria ser una funcionalitat atractiva de cara a futures ampliacions al projecte.

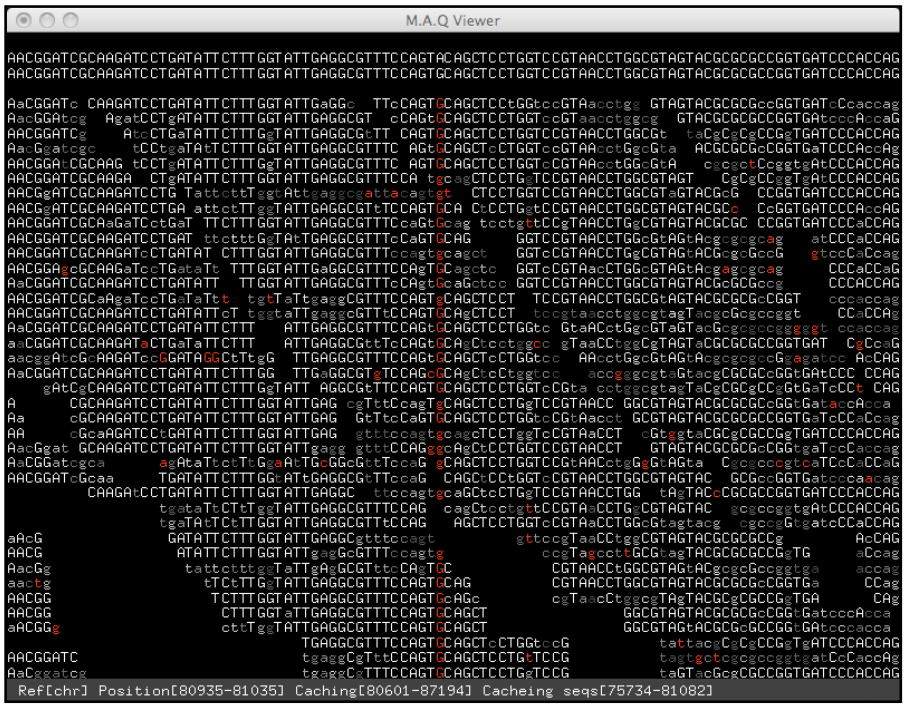


Figura 5-7. Aplicació MAQView del paquet MAQ

- **Resultats de les execucions**

MAQ tot i ser un algoritme orientat a un entorn d'execució de node únic ha sapigut estar en ple ús fins a l'actualitat en gran part per la seva qualitat d'alineament i per aconseguir uns temps respectables malgrat les pròpies limitacions d'aquest entorn. Per aquesta experimentació MAQ ha aconseguit mapejar el **94,2 %** de les **mostres mapejables**, és a dir *33/35* de les mostres esperades. Pels casos de Unmap, és a dir, pels que no mapejen el genoma, cal dir que MAQ ha aconseguit detectar els 15 casos dels 15 totals, és a dir el **100%** del total de les mostres **no Mapejables**.

MR MAQ amb el cas del **HIT** ha aconseguit detectar el **85,7%** de les **mostres mapejables**, és a dir *30/35* i el **100%** de les **no Mapejables** (en cap moment de l'execució ha imprimit els identificadors de les mostres no mapejables a la sortida).

Per tant podem dir que l' algoritme funciona adequadament amb uns valors d'eficiència en el mapeig molt similars a MAQ amb només amb un **8,5%** de marge d'error més pels casos de **HIT**. Aquest marge d'error s'investigarà i s'analitzarà el perquè a l'apartat de conclusions i millores.

5.3 Bloc 2: Proves en HPC

Un cop s'ha comprovat que l'algoritme funciona correctament en un entorn d'execució controlat i amb mostres acotades és moment d'avaluar altres objectius més complexos propis dels entorns multicluster. Per aconseguir-ho en aquestes proves es treballarà amb dades reals, és a dir, cromosomes sencers i amb conjunts de mostres short·Reads reals amb un tamany de dades entre les 20.000 i 100.000 mostres.

5.3.1 Cromosoma 22 sencer i conjunt de mostres real

Per aquesta experimentació s'ha treballat amb el cromosoma 22 real del genoma humà que té un tamany total aproximat de **40 x 10^6 bps** i amb un conjunt de mostres de gran qualitat facilitat per la companyia **Solexa-Ilumina**.

Per poder fer una experimentació formal i acurada hem trobat important primer realitzar les proves amb un únic node i amb una única fase Map i Reduce. D'aquesta manera podrem estudiar el seu rendiment i així determinar els valors òptims de sintonització per escalar-ho al cluster.

- **Proves amb 1 Node: 1 Map , 1 Reduce, 3 Gaps, 36 bps per les mostres.**

Anem a avaluar els temps obtinguts per a una unitat per així poder comparar amb altres configuracions i mesurar eficientment el rendiment de l'aplicació. Els resultats d'aquesta execució són:

Fase	Temps Exec. (seg)
Map	201
Reduce	277

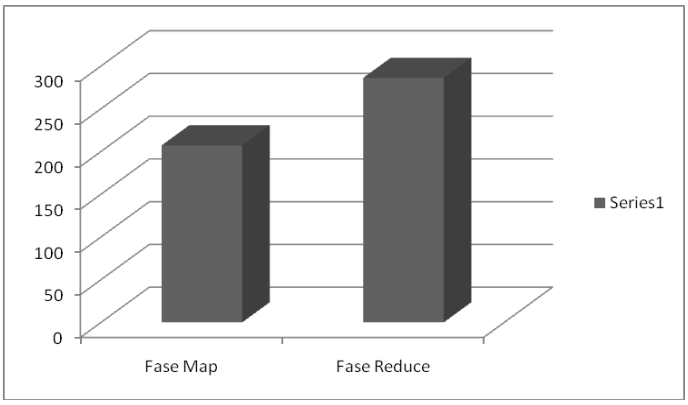


Figura 5-8. Gràfica t.Execució per Fases

Recordem que el Reduce estava format per diferents etapes: **Shuffle, Sort** i la pròpia **funció Reduce** i és precisament per aquest motiu que el temps total d'execució no és tracta de la suma directa de la fase de Map i de Reduce (en realitat la etapa de shuffle s'executa paral·lelament a la de Map). Aleshores si desglossem per etapes obtenim:

Fase	Temps Exec. (seg)
Map	201
Shuffle	195
Sort	7
Reduce cpu	75

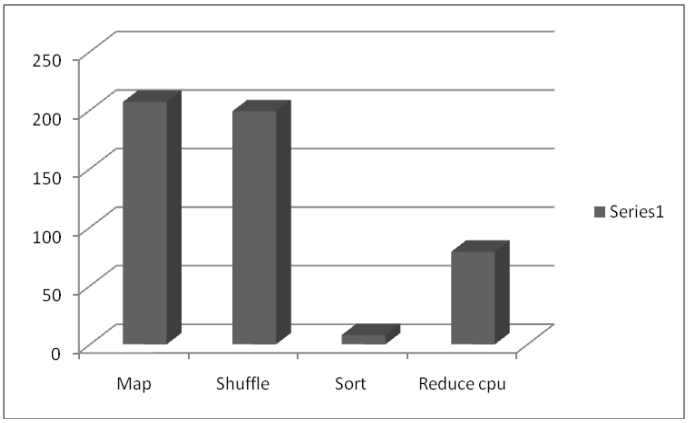


Figura 5-9. Gràfica T.Execució amb Fase Reduce desglossada

Podem observar com **la fase més costosa en quant a temps d'execució és la fase completa de Reduce** (shuffle, sort i Reduce cpu). Aquesta fase també coincideix en ser la part més costosa en quant a demanda de recursos, que en aquest cas representa la **CPU**. Això s'explica ja que, com hem pogut estudiar en capítols anteriors, la fase Reduce és la encarregada de fer tots els **càlculs matemàtic**s i operacions importants d'un conjunt d'entrada donat. Aquest conjunt d'entrada (grups de tuples amb la mateixa clau) recordem que és generat pel propi node local que ha executat el Map (**guarda el resultat a RAM**) i representa totes les dades necessàries per executar la funció, és a dir, la funció Reduce no precisarà de dades extres per al seu càlcul.

Tot el contrari passa amb la **fase Map**, la qual té un **potencial de càlcul molt baix** però una **demanda de dades** per completar la seva funcionalitat **molt gran**. Es per aquest motiu el qual la fase de Map està gran part del seu temps en una fase d'espera, és a dir, en **temps de E/S.** Això s'explica ja que la funció Map requereix de dades emmagatzemades al **HDFS** el qual està muntat de forma lògica en els discs durs del **HPC**. Per aquest motiu el número **d' accessos a discs durant la fase Map és molt elevat** i el temps d'espera també.

Amb aquesta informació és fàcil veure com la fase amb un paral·lelisme més immediat és la etapa de Map. D'aquesta manera podrem **executar varios Maps concurrentment** per cada node i així aprofitarem aquests temps d'E/S que es produeix. En quant a la fase de Reduce el paral·lelisme per node és més crític ja que pràcticament no existeix temps de E/S, essent la majoria temps d'execució a la CPU. Tot i així el que si podem fer és aprofitar que tenim nodes de doble nucli per executar 2 reduces per node, és a dir, **1 reduce per cada core**.

En aquest punt ens trobem amb un **problema amb el Framework de Hadoop**. *Hadoop,* al igual que molts frameworks i **middlewares**, estableix una **capa d'abstracció** important molt útil per algunes coses però que a la mateixa vegada ens **impedeix monitoritzar valors de baix nivell** de primera mà (memòria RAM necessària per cada fase, el temps dedicat al Read/Write de disc, el temps perdut en l'intercanvi de dades degut al tràfic de la xarxa etc.). És per això que l'ús de monitors com **time()** per sistemes *Unix* o classes Java com el **JProfiler** no són vàlids per aquest paradigma. Aquest factor ens suposa un dels pocs aspectes negatius que hem trobat al Framework (els monitors de Hadoop jobTracker i Tasktracker són massa generals i no permeten monitoritzar amb precisió valors de baix nivell importants com els anteriors esmentats) i suposa també una futur aspecte a tenir en compte per ampliacions per al projecte.

Malgrat els problemes d'aquesta gran abstracció hem continuat amb el treball de sintonització i hem establert a partir d'eines externes (com la de **top** de Unix) uns **paràmetres empírics òptims** per a la nostra aplicació i per als conjunts de dades utilitzats. Aquests paràmetres entre d'altres, estableix com a millor configuració pel Map la de **4 Maps per node** i de **2 Reduces per Node.** Amb aquesta configuració s'aconsegueix **contrarestar el temps d'E/S del Map** per aprofitar-ho en temps de CPU per altres Maps i aprofitar la **potència dels sistemes dual Core** per l'execució de la etapa Reduce.

Amb aquesta sintonització (i altres paràmetres d'administrador correctament sintonitzats) observem com ara el temps ha millorat molt considerablement.

Fase	Temps Exec. (seg)
Map	152
Shuffle	147
Sort	3
Reduce cpu	31

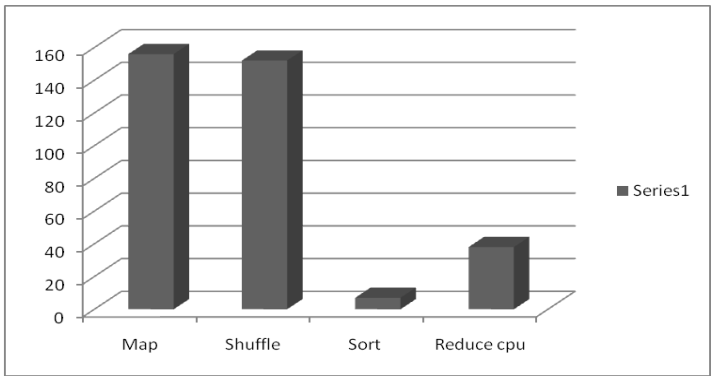


Figura 5-10. Gràfica T.Execució amb Fase Reduce desglossada amb la sintonització del framework optimitzada

Amb la sintonització el map ha incrementat el seu rendiment en un **24,37%,** el Shuffle un **24,61%,** el sort un **57,14%** i el Reduce un **58,66%.** És una configuració per tant molt positiva i que es guardarà per les següents proves del capítol.

- **Número de Mostres**

Per aquesta experimentació hi ha dos paràmetres clau que influeixen en la complexitat del problema: el tamany del fitxer de mostres short·Reads i el número de Gaps seleccionats. Veiem doncs en números la influència d'aquestes paràmetres :

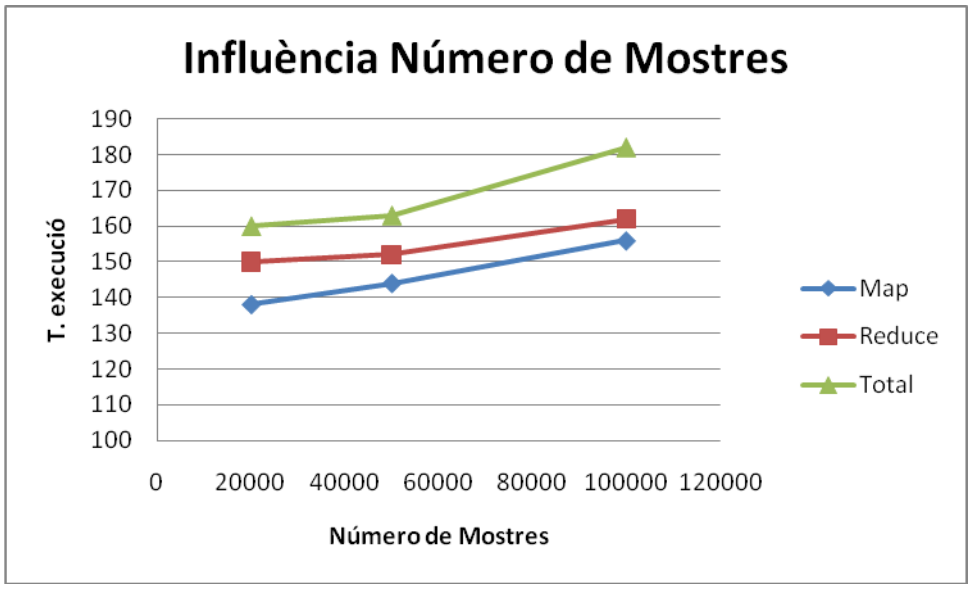


Figura 5-11. Gràfica influència número mostres

Podem observar com a mesura que ampliem el número de mostres short·Reads el temps d'execució del Map creix d'una **forma lineal constant**. Aquesta conclusió és evident ja que a més mostres la funció Map haurà de crear més tuples. En canvi al Reduce veiem que no sempre l'augment del tamany del fitxer de mostres es tradueix en molt més temps d'execució. Això ho veiem amb el canvi de 20.000 a 50.000 mostres en el que tot just augmenta dos segons el temps d'execució. Això es degut a que el Reduce, a diferència del Map, pot no incrementar la seva complexitat a causa d'haver gran quantitat de mostres que no mapejin el genoma. Aquest factor dependrà en gran mesura de les mostres i del genoma de referència que utilitzem. Amb l'augment cap a les 100.000 veiem que el grau de mapeig augmenta i per tant també la complexitat del reduce.

- **Número de Gaps**

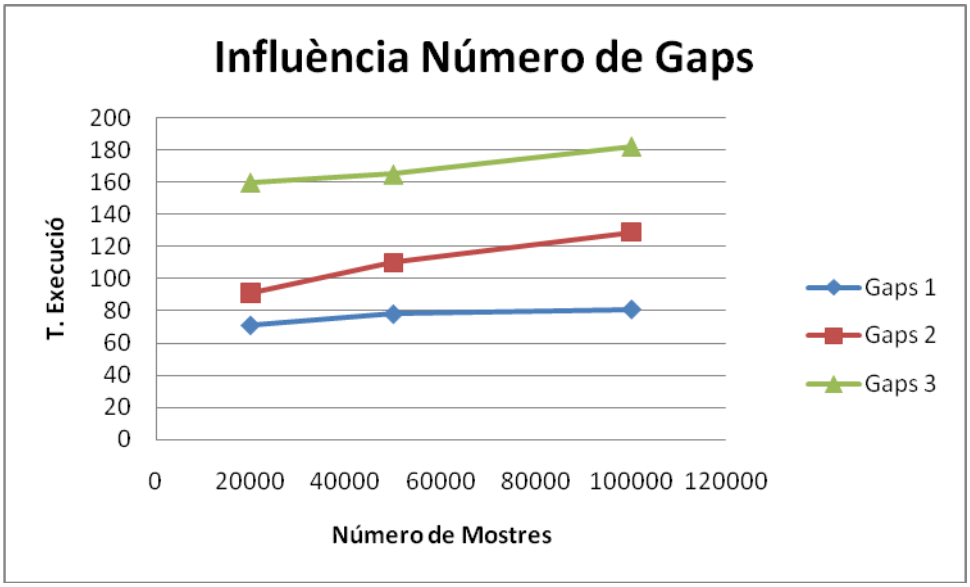


Figura 5-12. Gràfica influència número gaps

Amb un i dos Gaps veiem que la complexitat de l'algoritme per 20.000 i 50.000 mostres no dista gaire. Diferent és quan treballem amb 100.000 mostres on l'ús d'un Gap o de dos ja dista en una diferència de 42 segons. Aquest augment encara és veu més clar quan treballem amb el màxim número de Gaps, és a dir tres. Amb la selecció de tres Gaps veiem com la complexitat és dispara i és situa en pràcticament en el doble de la complexitat de l'ús d'un sol gap. Aquest comportament s'explica ja que cal recordar que la nostra aplicació amb l'opció de tres gaps està permetent una flexibilitat entre genoma de referència i mostres de 3 forats, és a dir, hi haurà possibilitat de que mapejin mostres amb un índex d'error de 3 bases per mostra, una opció que dispara les dades generades pel map d'una forma lineal i que naturalment afecta a la complexitat de l'algoritme.

5.3.2 Proves al Cluster

Arriba el moment d'escalar les proves al cluster per estudiar el comportament de l'algoritme per entorns multi-node. Aquestes proves són molt importants ja que amb elles podrem comprovar l'assoliment d'objectius com: : **l'escalabilitat, rendiment, speed-up, eficiència** etc.

Per a la experimentació treballarem amb un cluster dedicat d'un màxim de 4 nodes amb les especificacions que es van comentar al principi del capítol. Per mesurar els temps amb els diferents recursos s'han pres mesures d'una forma lineal amb un, dos i quatres nodes respectivament.

A la **figura 5-13** podem veure mitjançant el monitor de Hadoop, el **jobtracker,** com el nostre cluster està muntat amb èxit i a l'espera del llançament de treballs. Amb el monitor podem veure detalls del nostre cluster : quatre nodes vius amb els noms de **AOPCSO5, AOPCSO8, slave4 i slave6;** i detalls del **HDFS** que en aquest cas té una capacitat màxima de **57,67Gb** (valor obtingut de la suma lògica de tots els disc durs dels nodes del cluster : **15 gb x 4 ≈ 57,65gb**)

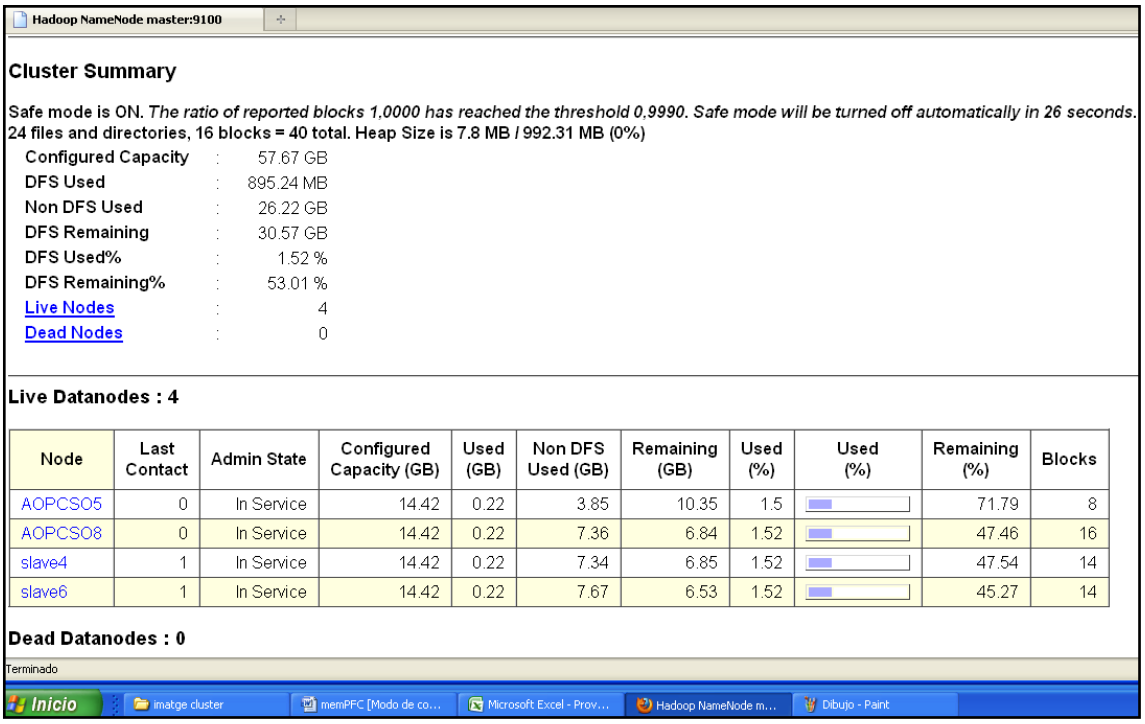


Figura 5-13. Monitor **jobtracker**, ens dona detalls del HPC

Les proves al HPC s'han fet amb la totalitat de les mostres (100.000), el cromosoma 22 sencer i amb el màxim número de Gaps (3). Els resultats són els mostrats en la següent taula (expressada en segons):

Nodes	Map	Reduce cpu	Total	Teòric
1	152	31	183	183
2	62	15	88	91,5
4	30	9	54	45,75

i en forma de gràfica:

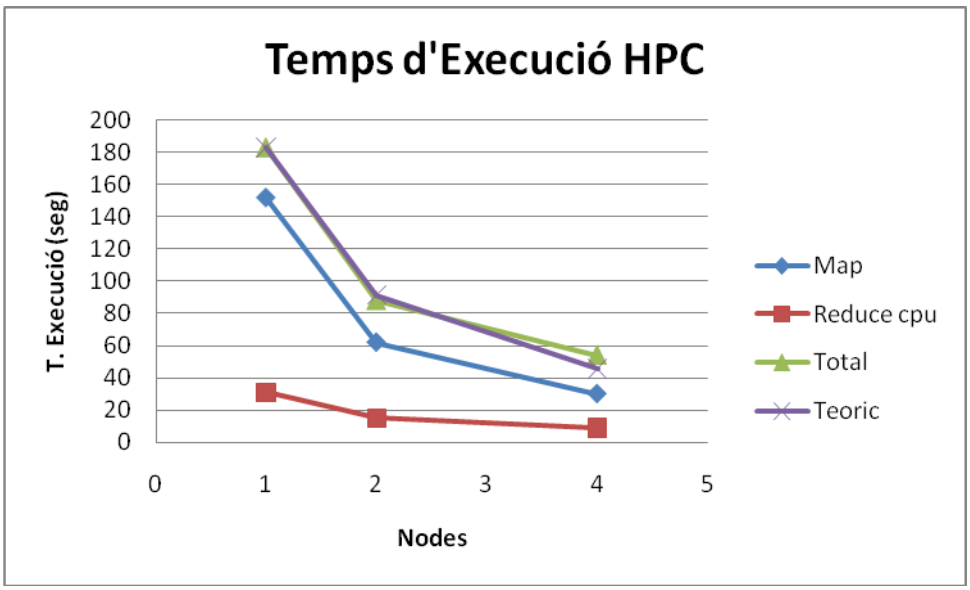


Figura 5-14. Gràfica t.Execució al cluster

Tant els valors de la taula com la gràfica mostren una linealitat clara per a totes les fases del treball . Fins i tot una linealitat millor de la esperada en quant al temps d'execució. Observem la **fase Map**; el temps amb un node és de 152 segons motiu pel qual s'esperaria que amb dos nodes s'aconseguís un temps teòric ideal aproximat de 76 segons (152/2). Veiem però que amb dos nodes s'aconsegueix millorar encara més aquest temps i l'algoritme ha obtingut un temps de 62 segons. El mateix passa pel temps de 4 nodes que s'esperaria un temps de 38 segons (152/4) però s'aconsegueix un temps de 30 segons. Són uns valors molt positius i que expliquen la **forma lineal pronunciada** de la gràfica.

Per a la **fase Reduce** passa exactament el mateix. Entre l'execució d'un i dos nodes veiem com es compleix la mateixa condició que pel Map. Només és veu abaixat amb l'execució amb 4 nodes en la qual s'esperaria un temps de 7,75 segons (31/4) i s'aconsegueix un temps de 9 segons. Tot i així és un resultat molt proper al valor teòric ideal i que també té una tendència lineal pronunciada molt clara i eficient.

Finalment en quant al **temps total d'execució** comentar també que lògicament segueix la mateixa tendència del Map i del Reduce. Els temps de **183 i 88** són millor dels esperats però amb l'últim temps (**54 segons**) i a causa de l'execució en 4 nodes del Reduce veiem que perd uns segons respecte els **45,75 segons** del temps teòric ideal (183/4). Aquest fet pot ser donat a que *Hadoop* utilitza el **protocol HTTP** per enviar les dades al cluster. Com sabem aquest protocol, *Internet,* pateix d'un cert retard o **"lag"** que molt probablement ens està afectant en bastants segons. A més aquest lag és major quants més nodes afegim al **cluster**, factor que s'explica ja que són més

dades les que cal enviar per la xarxa. Amb aquestes dades som capaços de generar una gràfica que ens deixa veure més clars aquests valors:

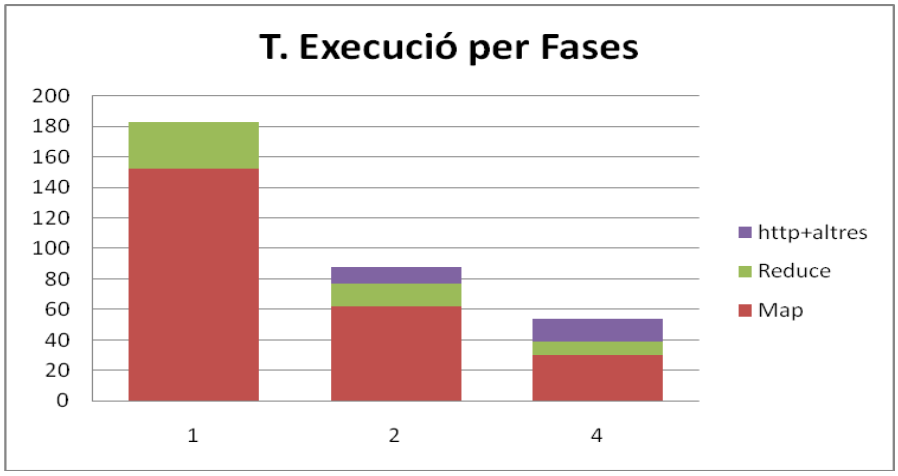


Figura 5-15. Gràfica t.Execució per Fases amb soroll

Efectivament amb la gràfica veiem com **a més nodes involucrats al HPC més gran és el temps de latència** d'espera a la xarxa. Això ens dona una bona pista cara a factors a millorar per possibles ampliacions a aquest projecte. Possiblement amb una **implementació local o de nivell 2 a la capa OSI** on s'utilitzés un switch per comptes d'un router (**de nivell 3**) s'obtindria millors resultats. Tot i així es pot veure com els resultats de totes les fases donen uns números molt positius, propers tots als valors teòrics ideals. Aquestes conclusions però és veuran més clares i reforçades amb els estudis següents del **speed-up** i la **eficiència**.

- **Speed-up al cluster**

El **Speed-up** és el paràmetre que mostra com d'eficient resulta el paral·lelisme en un algoritme respecte l'execució convencional en un node de forma seqüencial. És a dir, per al nostre problema el *Speed- up* mostrarà el rendiment, o guany, entre l'execució amb un node i l'execució amb *n* nodes. Aquest s'expressa de forma:

$$Speed\text{-}Up = \frac{T\_Execució\_amb\_1\_node}{T\_Execució\_amb\_N\_nodes}$$

D'aquesta manera obtenim que el *Speed-up* per a la nostra aplicació és:

Nodes	SpeedUp
1	1
2	2,07
4	3,38

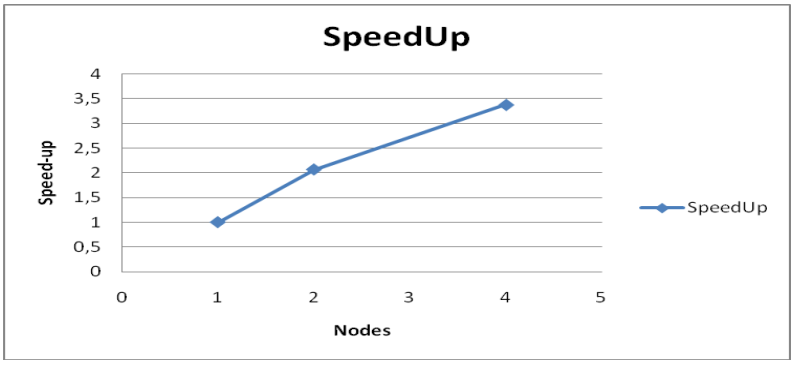


Figura 5-16. Gràfica SpeedUp

Com podem veure el speed-up és lineal aspecte que significa que l'aplicació té un rendiment molt alt. En quant al tipus de linealitat, més pronunciada entre l'execució d'un node i de dos, es deu a, com ja hem fet menció anteriorment, que l'execució amb un node i dos és millor del esperat i això fa que la línia sigui més pronunciada. Però aquest cas com és lògic no és negatiu sinó ben bé tot el contrari.

- **Eficiència al HPC**

La eficiència és un paràmetre fortament lligat al *speed-up* però té en consideració el número exacte de recursos que s'han empleat per cada experimentació. És per aquest motiu que les dades de la eficiència són més sensibles a les inilinealitats i mostren millor el guany en afegir més nodes a l'aplicació. Aquesta és defineix:

$$Eficiència = \frac{T\_Execució\_amb\_1\_node}{T\_Execució\_amb\_N\_nodes \times N\_nodes}$$

Per a la eficiència veiem oportú desglossar-ho en les diferents fases de l'algoritme Map i Reduce. Els valors obtinguts per aquestes són:

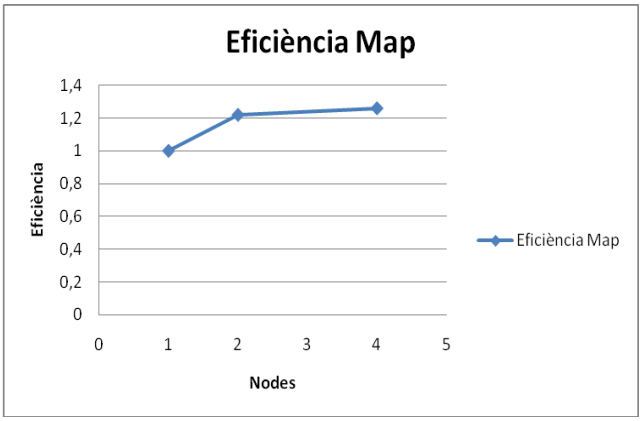


Figura 5-17. Gràfica Eficiència Map

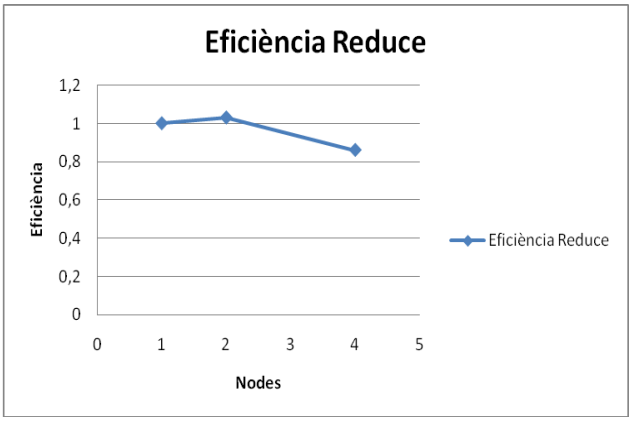


Figura 5-18. Gràfica Eficiència Reduce

Podem observar com a la fase Map l'eficiència és excel·lent. A majors recursos afegits millor és el rendiment obtingut. Això vol dir que la fase Map té un índex de paral·lelisme immillorable. Amb el Reduce no passa el mateix. Aquesta fase tot i tenir una eficiència prou bona apreciem com arriba un punt en que el paral·lelisme no és tant eficient com al Map. Això era d'esperar ja que la fase Reduce és bastant més complexa que la de Map. Aquesta baixada en la eficiència del Reduce afectarà molt probablement a la eficiència global de l'aplicació la qual és la suma de totes dues fases (no directa). Observem la gràfica global:

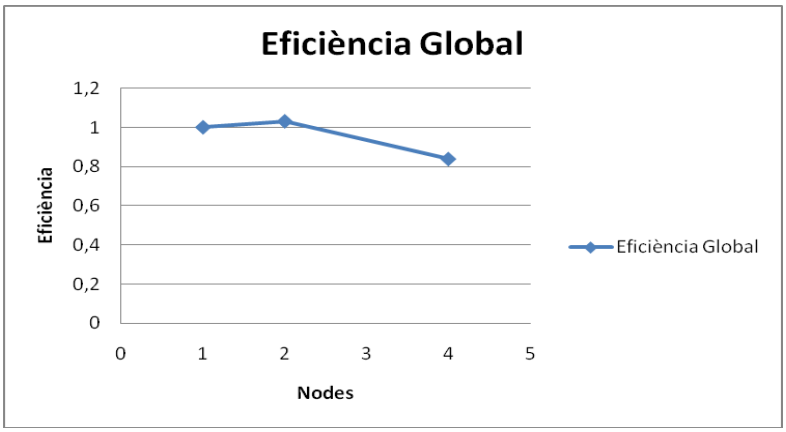


Figura 5-19. Gràfica Eficiència Global

Efectivament ocorre el que nosaltres esperàvem: l'eficiència global abaixa lleugerament per l'execució amb quatre nodes. Seria interessant que en futures ampliacions es fes l'experimentació amb més nodes per veure si aquesta eficiència remunta o definitivament continua baixant. Tot i així **podem afirmar que l'aplicació MR MAQ és eficient i queda demostrat el seu correcte funcionament per a entorns paral·lels com el de l'experimentació**.

- **Balanceig al cluster**

Finalment i com a últim aspecte a tenir en compte per aquesta experimentació, observarem el **paràmetre del balanceig**. Aquest paràmetre és clàssic en **entorns HPC** i mostra el **nivell de treball** que ha sofert **cada node**. El balanceig en un HPC és crític ja que un cluster mal balancejat podria repercutir molt negativament en el rendiment de l'aplicació i es traduiria amb una càrrega de treball diferent per cada node. Afortunadament el **framework de Hadoop** també **s'encarrega** d'aquest aspecte i ho fa **molt eficientment.** Amb **Hadoop** cada node rep una càrrega de treball molt similar a la dels altres i assegura un cop més un rendiment òptim. Observem-ho a les diferents figures:

1. Balaceig Map

El jobtracker de Hadoop ens permet estudiar el grau de balanceig del cluster per fases. Per aquesta experimentació com sabem s'ha executat amb el màxim de nodes, és a dir 4, on s'han corregut un total de 16 tasques de Map i 8 de Reduce. Com veiem a les **figures 5-20.1 i 5-20.2** totes les tasques Map tenen un volum de feina entorn els 10 segons aproximadament. **De manera que podem afirmar que tots els nodes reben una càrrega de treball molt similar i que per tant el cluster està correctament balancejat per aquesta fase**.

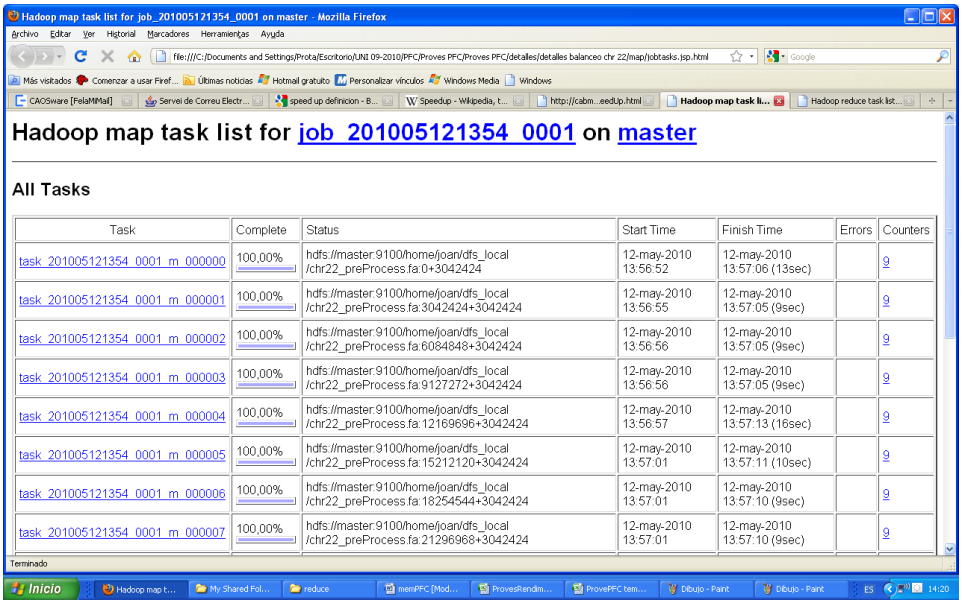
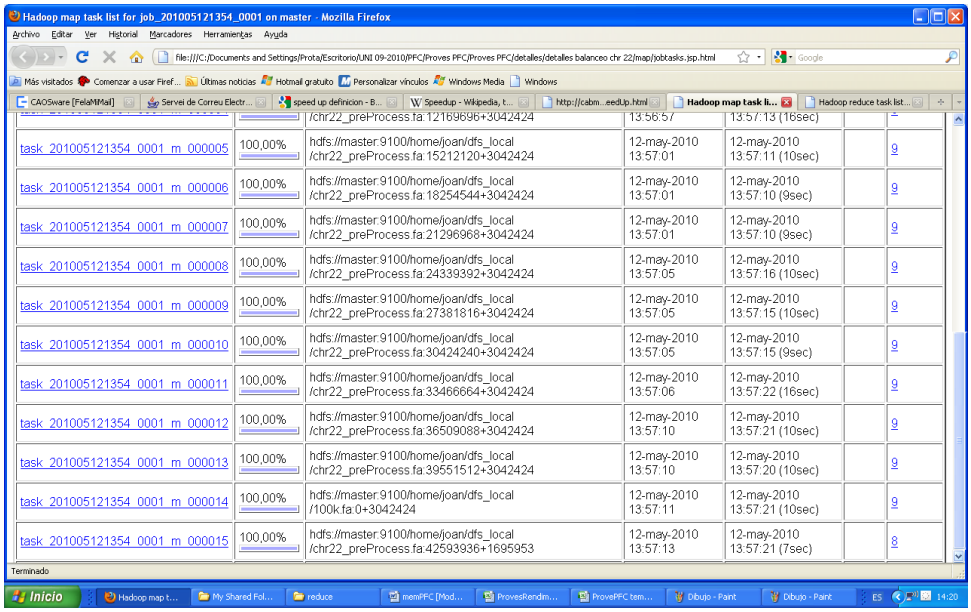


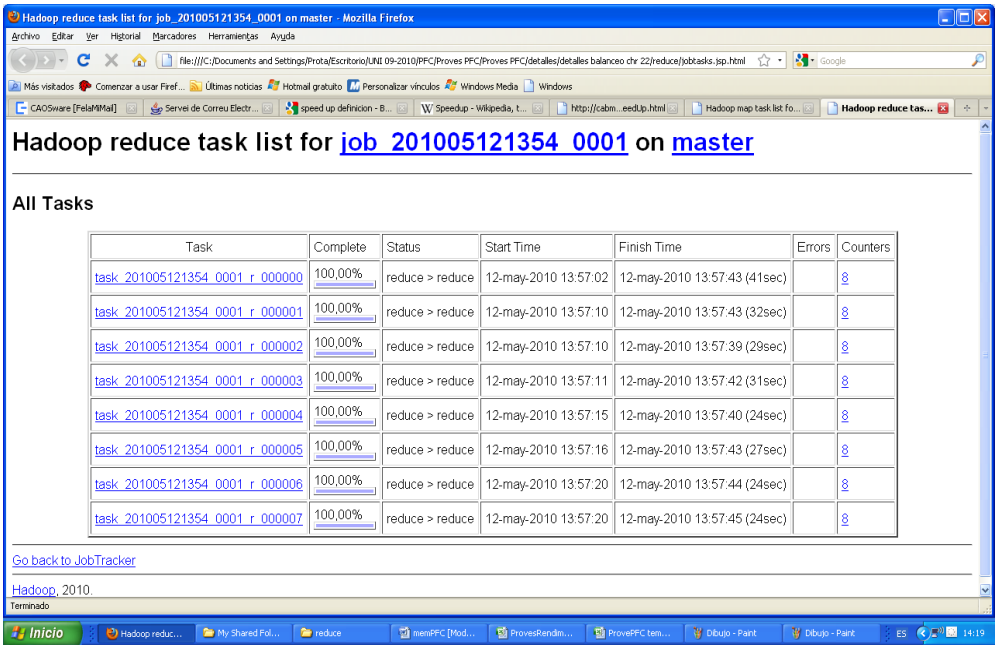
Figura 5-20.1. Balanceig al cluster Fase Map (Map 1-8)



task_201005121354_0001_m_000005	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:15212120+3042424	12-may-2010 13:57:01	12-may-2010 13:57:11 (10sec)	9
task_201005121354_0001_m_000006	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:18254544+3042424	12-may-2010 13:57:01	12-may-2010 13:57:10 (9sec)	9
task_201005121354_0001_m_000007	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:21296968+3042424	12-may-2010 13:57:01	12-may-2010 13:57:10 (9sec)	9
task_201005121354_0001_m_000008	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:24339392+3042424	12-may-2010 13:57:05	12-may-2010 13:57:16 (10sec)	9
task_201005121354_0001_m_000009	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:27381816+3042424	12-may-2010 13:57:05	12-may-2010 13:57:15 (10sec)	9
task_201005121354_0001_m_000010	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:30424240+3042424	12-may-2010 13:57:05	12-may-2010 13:57:15 (9sec)	9
task_201005121354_0001_m_000011	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:33466664+3042424	12-may-2010 13:57:06	12-may-2010 13:57:22 (16sec)	9
task_201005121354_0001_m_000012	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:36509088+3042424	12-may-2010 13:57:10	12-may-2010 13:57:21 (10sec)	9
task_201005121354_0001_m_000013	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:39551512+3042424	12-may-2010 13:57:10	12-may-2010 13:57:20 (10sec)	9
task_201005121354_0001_m_000014	100,00%	hdfs://master:9100/home/joan/dfs_local /100k.fa:0+3042424	12-may-2010 13:57:11	12-may-2010 13:57:21 (10sec)	9
task_201005121354_0001_m_000015	100,00%	hdfs://master:9100/home/joan/dfs_local /chr22_preProcess.fa:42593936+1695953	12-may-2010 13:57:13	12-may-2010 13:57:21 (7sec)	8

Figura 5-20.2. Balanceig al cluster Fase Map (Map 8-16)

2. Balanceig Reduce



Task	Complete	Status	Start Time	Finish Time	Errors	Counters
task_201005121354_0001_r_000000	100,00%	reduce > reduce	12-may-2010 13:57:02	12-may-2010 13:57:43 (41sec)		8
task_201005121354_0001_r_000001	100,00%	reduce > reduce	12-may-2010 13:57:10	12-may-2010 13:57:43 (32sec)		8
task_201005121354_0001_r_000002	100,00%	reduce > reduce	12-may-2010 13:57:10	12-may-2010 13:57:39 (29sec)		8
task_201005121354_0001_r_000003	100,00%	reduce > reduce	12-may-2010 13:57:11	12-may-2010 13:57:42 (31sec)		8
task_201005121354_0001_r_000004	100,00%	reduce > reduce	12-may-2010 13:57:15	12-may-2010 13:57:40 (24sec)		8
task_201005121354_0001_r_000005	100,00%	reduce > reduce	12-may-2010 13:57:16	12-may-2010 13:57:43 (27sec)		8
task_201005121354_0001_r_000006	100,00%	reduce > reduce	12-may-2010 13:57:20	12-may-2010 13:57:44 (24sec)		8
task_201005121354_0001_r_000007	100,00%	reduce > reduce	12-may-2010 13:57:20	12-may-2010 13:57:45 (24sec)		8

Go back to JobTracker

Hadoop, 2010.

Figura 5-21. Balanceig al cluster Fase Reduce

El mateix passa per a la Fase Reduce la qual veiem s'ha executat en tots els nodes amb un temps d'execució entre els 45 i 25 segons. Per tant **podem afirmar que el cluster està correctament balancejat també per a la Fase Reduce**.

Capítol VI

Conclusions

Ha arribat el moment de fer balanç sobre els resultats obtinguts i de mesurar de la forma més objectiva possible el compliment dels objectius inicials marcats. A més a més, al final d'aquest capítol, també farem menció d'algunes possibles línees de continuïtat per aquest Projecte.

6.1 Compliment dels Objectius

En tot gran Projecte sempre s'espera l'assoliment de molts i diferents objectius amb el desig d'acomplir sinó tots, el major nombre possible. En el nostre cas no va ser diferent i vam enumerar una llarga llista d'objectius i subobjectius. Si recordem aquests eren els següents:

- **Realitar un estudi general sobre els conceptes de Biologia i Genòmica. Aprendre exercicis bàsics de seqüenciació genòmica.**

  Aquest objectiu bàsic podem afirmar que ha sigut **complert amb èxit** i que va resultar fonamental per a la implementació de l'algoritme de Read Mapping final. Amb aquesta aproximació al tema vam poder començar a endintrar-nos en importants conceptes de genòmica necessaris per a la realització d'aquest Projecte.

- **Realitar un estudi profund** d'algoritmes genòmics de **Read Mapping** i estudiar **l'aplicació** del framework de **Hadoop** en aquest tipus d'aplicacions.

  Aquest objectiu, estimat amb una duració total de 10 dies, va ser **complert correctament però va patir d'una retallada** a poc més de dos dies de treball. La raó va ser que amb el director vam considerar més oportú estudiar els algoritmes després d'adquirir una base amb el Framework de Hadoop.

- **Realitar un estudi profund** per tal d'adquirir una bona formació i experiència amb el **framework** MapReduce de **Hadoop.**

  Aquest objectiu també va ser **complert** a la fase inicial del Projecte amb les proves de l'aplicació *WordCount*. **Tot i així cal reconèixer que l'aprenentatge important,** en quant a la formació al Paradigma de Programació Map Reduce, **va produir-se durant la pròpia implementació de l'algoritme**.

- Estudiar dos algoritmes genòmics importants de Read Mapping : **CloudBurst** i **MAQ** i **implementar un algoritme propi de Read Mapping** orientat per funcionar en un *cluster* de multiples nodes.

  Aquesta tasca, tot i que va ser estimada amb una llargada considerable (40 dies), cal dir que no ho va ser lo suficient i que **és va ampliar a pràcticament als 60 dies de treball.** La raó va ser donada a la complexitat dels algoritmes i a la inexperiència amb la programació del Paradigma Map Reduce. Tot i així l'hem de considerar un **objectiu complert**.

- **Proves i depuració de la primera implementació.**

  Aquest objectiu, tot i que inevitablement desplaçat, va ser **complert** sense problemes i per a una durada molt equivalent a la prevista.

- **Proves en HPC i sintonització.**

  És important dir que **aquest objectiu va** fer **recuperar en part l'objectiu anterior** (depuració de la primera implementació). Això va ser degut a que l'algoritme havia sigut desenvolupat i pensat per a un entorn Multi-Node però fins al moment només havia sigut provat per a un entorn de Node únic (menys complexe). D'aquesta manera al ser avaluat en un **HPC** *real* es va fer necessari tornar a fer algun canvi en el codi per a obtenir un correcte funcionament.

Aquestes **pujades de temps no previstes** en l'evolució del Projecte **i la falta d'experiència per trobar un monitor capaç d'avaluar eficientment el rendiment del Framework de Hadoop** van ocasionar que el segon subobjectiu (sintonització de l'entorn) no pogués ser tan exhaustiu com ens hagués agradat i representa per tant un **objectiu no complert.**

- **Objectius al HPC : Balanceig, Speed-Up, Eficiència *i* Escalabilitat.**

  Amb les experimentacions realitzades vam poder comprovar de primera mà com **Hadoop balancejava excel·lentment** totes les subtasques de cada treball. També gràcies a les diferents proves vam veure com els temps d'execució en alguns casos eren fins i tot millors dels temps teòrics esperats i per això el **Speed-Up** era tan **bo**. En quant a la **eficiència** vam poder apreciar com era **molt positiva per a la fase de Map** però **millorable per a la de Reduce**. Finalment, i amb tots els números obtinguts, podem afirmar que l'algoritme escala correctament. Això és una raó importantíssima per estar satisfets ja que representava un dels objectius principals per aquest Projecte.

- **Proves a gran Escala.**

  Hauria estat molt interessant provar l'algoritme en un HPC de **major nombre de nodes**. Malauradament la falta de temps ha fet que l'acompliment d'aquest objectiu s'hagi **resolt només parcialment** (s'ha experimentat amb un número màxim de 4 nodes).

- **Redacció de la memòria.**

  Aquest objectiu, i aquestes línies són bona prova, s'ha **complert amb èxit** i segons les dates previstes (tot i que també s'ha allargat considerablement del temps previst). El director de Projecte em va aconsellar fer de la redacció de la memòria una tasca continua i no puntual i creiem que el resultat ha sigut molt positiu.

6.2 Ampliacions Proposades i Optimitzacions

En diversos capítols de la memòria s'ha fet menció d'algunes observacions del Projecte i de possibles optimitzacions del propi funcionament. Aquestes millores representen línies de continuïtat que en un futur podrien transformar-se en altres Projectes complementaris a aquest. Aquestes línies futures les classificarem en dos grups: les possibles millores i les funcionalitats extres de l'aplicació.

- **Possibles millores**

1. Un handicap important amb el que ens vam trobar amb el Framework de Hadoop és la falta d' un monitor eficient capaç de mesurar dades de baix nivell com ara el I/O Bound, la memòria consumida per cada fase, el nivell d'accès a disc etc. Per internet hem vist que existeixen alguns monitors genèrics com ara Hyperic o Ganglia que afirmen ser capaços de mesurar aplicacions programades amb el Framework de Hadoop. Seria per tant interessant estudiar aquestes aplicacions i provar-les amb el nostre algoritme.

2. Al capítol d'Experimentació vam poder apreciar com la fase Map de l'aplicació era força millor a la de Reduce. Aquest fet podria ser degut, entre altres qüestions, al tràfic que patia la xarxa en aquells moments. Per tant una altra millora seria muntar el cluster sobre una xarxa mitjançant un Switch el qual administrés la connexió en un àmbit local.

3. A l'experimentació que es comparava l'aplicació amb MAQ vam poder veure com hi havia una certa diferència a l'hora de detectar els casos positius o de Hit (un 8,5%). Aquest fet el vam estudiar posteriorment i vam determinar que era degut a la funció de pre-processament de l'algoritme. Aquesta funció divideix el fitxer del genoma de referència d'entrada per tal de fer-ho compatible amb el tipus de programació del paradigma Map Reduce. Aquesta conversió però, pot ocasionar que en alguns casos molt puntuals les particions es fagin en zones directes de Hit i que per tant es perdin alguns casos de Hit. Seria per tant doncs interessant també fer un estudi més profund d'aquesta part del algoritme i acurar-la per a tenir un resultat encara més òptim.

- **Funcionalitats Extres**

1. Seria molt interessant que en ampliacions futures s'avalués l'aplicació en **clusters de gran tamany entorn els 100 i fins i tot els 1.000 nodes**. També, i aprofitant el major tamany dels HPC, es podria experimentar amb varios genomes a la vegada i amb una quantitat de mostres short·Read molt superior.

2. Implementar tota una **interfície gràfica** que fes més còmode l'execució amb l'aplicació. Aquesta ampliació seria relativament fàcil ja que el Projecte està desenvolupat integrament en Java i per tant tindria una ràpida integració amb les llibreries Swing i AWT (biblioteques gràfiques de Java).

3. Una altra possible ampliació que resultaria molt interessant d'implementar seria una aplicació semblant a la de **MAQView** de *MAQ* per veure de forma gràfica els resultats obtinguts per l'aplicació com ara el nivell d'alineament i de solapament.

Referències

A continuació es detallen les referències principals per aquest Projecte:

[1] *Apache Hadoop Org*. Web oficial del Projecte Hadoop de Apache., **http://hadoop.apache.org/**

[2] *Apache Hadoop Org*. Api Hadoop 0.20.2 de Apache., **http://hadoop.apache.org/common/docs/current/api/**

[3] *Apache Hadoop Org*. Wiki i comunitat d'usuaris del Projecte Hadoop de Apache., **http://wiki.apache.org/hadoop/**

[4] *Apache Hadoop & Yahoo.* Tutorial Framework Hadoop, **http://developer.yahoo.com/hadoop/tutorial/module4.html**

[5] *Michael Schatz*. Host oficial i Paper del Algoritme CloudBurst, **http://sourceforge.net/apps/mediawiki/cloudburst-bio/index.php?title=CloudBurst**

[6] *Heng Li*. Host oficial i Paper del Algoritme MAQ, **http://maq.sourceforge.net/**

[7] *Michael G. Noll*. Manual d'instal·lació per a Single Node de Hadoop, **http://www.michael-noll.com/wiki/Running_Hadoop_On_Ubuntu_Linux_%28Single-Node_Cluster%29**

[8] *Michael G. Noll*. Manual d'instal·lació per al HPC de Hadoop, **http://www.michael-noll.com/wiki/Running_Hadoop_On_Ubuntu_Linux_(Multi-Node_Cluster)**

[9] *Comunitat Wikipedia*. Informació sobre la tècnica de compressió de Burrows Wheeler, **http://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform**

[10] *Comunitat Wikipedia*. Informació sobre la Seqüenciació i Re-Seqüenciació d'ADN, **http://en.wikipedia.org/wiki/DNA_sequencing**

[11] *Java Technology*. Informació diversa del llenguatge de programació Java, **http://java.sun.org**

Annexes

Tutorials Redactats de forma interna al departament

Instalando hadoop en debian
SingleNode Cluster

1. Cambiar/añadir user hadoop; (user sudo)

sudo passwd hadoop

2. Copiar archivos Map Reduce Platform. Cambiar permisos a todo el directorio;

sudo chown -R hadoop:hadoop

3. Instalar jdk

4. Instalar hadoop0.18.3.tar.gz

5. Setting variables de entorno;

HADOOP_INSTALL=/home/joan/Escritorio/Projects/hadoop_singleNode/

6. Crear symbolic link en hadoop; *ln -s hadoop0.18.3 hadoop*

7. Instalar vim (si es necesario). Editar y modificar hadoop-site.xml

```
<?xml version="1.0"?>
<?xmlstylesheet
type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/tmp/hadoop${
user.name}</value>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9100</value>
</property>
<property>
<name>mapred.job.tracker</name>
<value>hdfs://localhost:9101</value>
</property>
```

```
<property>
<name>dfs.replication</name>
<value>8</value>
</property>

<property>
<name>mapred.child.java.opts</name>
<value>Xmx512m</
value>
</property>
</configuration>
```

8. Asignar *JAVA_HOME= path_jdk/jdk1.6...revision_instalada en hadoop-env.sh*

9. Crear el sistema de ficheros distribuido (DFS) y formatearlo (sólo la primera vez que se instale);

mkdir -p /tmp/hadoopusername/dfs/name
$HADOOP_INSTALL/hadoop/bin/hadoop namenode format

10. Para evitar insertar password por cada operación con permisos setear al user como usuario de confianza.

su - hadoop
ssh-keygen -t rsa -P ""
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
ssh localhost

Instalando hadoop en debian MultiNode Cluster

1. Instalación simple de Single-Node Cluster.

2. Modificar /etc/hosts. Para ver ip's utilizar ifconfig o route

```
   # /etc/hosts (for master AND slave)

192.168.0.1 master
192.168.0.2 slave
```

3. Copiar public ssh key master al host slave:

(from hadoop@master); cat $HOME/.ssh/id_rsa.pub >>
$HOME/.ssh/authorized_keys (to hadoop@slave)

4. Configurar en el host master:

/conf/masters
/conf/slaves

5. Configurar en todos los hosts hadoopsite.
xml (igual para todos)

```
<property>
<name>fs.default.name</name>
<value>hdfs://master:9100</value>
<description>The name of the default file system. A URI whose
scheme and authority determine the FileSystem implementation. The
uri's scheme determines the config property (fs.SCHEME.impl) naming
the FileSystem implementation class. The uri's authority is used to
determine the host, port, etc. for a filesystem.</description>
</property>
<property>
<name>mapred.job.tracker</name>
<value>master:9101</value>
<description>The host and port that the MapReduce job tracker runs
at. If "local", then jobs are run inprocess
as a single map
and reduce task.
</description>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
<description>Default block replication.
The actual number of replications can be specified when the file is
created.
The default is used if replication is not specified in create time.
</description>
</property>
```

6. Formatear DFS desde el host master;

$HADOOP_INSTALL/hadoop/bin/hadoop namenode format

7. Iniciar daemons. Si todo es correcto deberiamos ver los procesos;

Master	Slave-i
NameNode	X
DataNode	DataNode
JobTracker	X
TaskTracker	TaskTracker

**Nota1: Para mirar los procesos utilizar /jdk/bin/jps*

8. Lanzar ejemplo wordCount, el mismo que para Single-Node cluster.

*Nota2 : Durante la instalación en los pasos 5,6 podemos encontrar errores en el slave de estilo *error on connecting : workspace name invalid.*En ese caso deberemos borrar todo el contenido de /tmp con user hadoop. Es un bug de la versión de hadoop.

Variables de Sintonització de Hadoop i com afecten

```
======================================== good

dfs.block.size
defecto (64mb): 67108864
nuevo (128mb):            134217728


======================================== good

io.sort.mb

defecto: 100
nuevo:  150

======================================== good

io.sort.factor

defecto: 10
nuevo:  12
```

```
====================================== good

io.sort.record.percent

defecto: 0.05
nuevo: 0.20

====================================== good

io.sort.spill.percent

defecto: 0.80
nuevo: 0.90

====================================== bad

mapred.compress.map.output
defecto: false
nuevo: true

====================================== good

io.file.buffer.size
defecto: 4096
nuevo: 131072

====================================== good

mapred.child.java.opts
defecto: -Xmx200m
nuevo: -Xmx256m


====================================== bad

io.file.buffer.size
defecto: 131072
nuevo:   200000
```

Pseudo-codi Algoriitme MAQ

1. Introducción

Este documento describe el funcionamiento básico del mapeador MAQ con el objetivo de establecer las bases para el desarrollo de una implementación usando hadoop como entorno de ejecución.

2. Funcionamiento básico de MAQ

La idea del funcionamiento de MAQ es bastante sencilla. El procesamiento se divide en dos etapas:

1. Indexar reads
2. Escanear el genoma de referencia para
    a. Identificar hits
    b. Extender y dar un valor de score a los hits encontrados

Un hit es una posición en el genoma de referencia donde hemos encontrado un read que esta potencialmente relacionado.

Específicamente, este funcionamiento se desarrolla de la siguiente manera:

Inicialmente, se definen seis plantillas para indexar las primeras 28 bases de los reads a procesar. Si las plantillas tuvieran 8 bits serían las siguientes:

I. 11110000 y 00001111
II. 11001100 y 00110011
III. 10101010 y 01010101

Cuando se aplica una plantilla a un read, se conserva la base que corresponde con el "1" y se pierde aquella que corresponde con el "0".

2.1. CONSTRUCCIÓN DEL ALINEAMIENTO

1: se leen todos los short reads en memoria

2: se aplica la primera plantilla a cada read

Ejemplo: ACGT...ACGT	read original
1111 ... 00000	plantilla
ACGT	resultado

A partir del resultado de aplicar la plantilla (14 bases) se genera un entero de 24 bits: Ni que se almacena en una lista junto con el identificador del read. Por lo tanto, la lista de procesamiento de los reads tiene esta forma:

(Ni, read_id), (Ni+1, read_id), ...

3: Se indexan los reads

Se ordenan los reads de la lista utilizando los enteros de 24 bits de forma que aquellos reads con el mismo Ni se almacenan agrupados en memoria. Típicamente, en forma de tabla hash con Ni como clave

4: Se escanea la referencia genómica base por base en subsecuencias de 28 bases en sentido directo e inverso (forward and reverse)

- Subsecuencia 28 bases
- Plantilla 11110000
- Obtenemos entero de 24 bits
- Buscamos en el hash de reads con el entero obtenido

- Posible hit encontrado

5: si encontramos un hit

Calculamos el número de mismatches entre la referencia y el read usando el read entero, alargando las subsecuencias iniciales de 28 bases, sin gaps

Guardamos los siguientes datos para el hit:

- Entero N de 24 bits
- Identificador de read read_id
- Posicion del hit en la secuencia de referencia
- Lectura directa o inversa
- Número de mismatches encontrados
-

6: usar templates 2 y 3 para repetir los puntos anteriores

7: Una vez generados todos los hits posibles, recorremos las posiciones donde se han encontrado estos hits y calculamos un valor de calidad para los hits con menor número de mismatches encontrados. A estos hits, aplicamos esta fórmula de calidad del alineamiento:

q1: mismatches del mejor hit
q2: mismatches del segundo mejor hit
n2: número de hits con el mismo número de mismatches que el segundo mejor hit
k': mínimo número de mismatches en la semilla de 28 bases
q: calidad promedio de las bases en la semilla de 28 bases
p(k,28): probabilidad de que, dada una secuencia de 28 bases, coexistan un hit perfecto y un mismatch de k bases

la calidad del mapping Qs se define como:

$$Qs = \min\{ q2 - q1 - 4.343 \log n2,\ 4 + (3 - k')(q - 14) - 4.343 \log p(3 - k', 28) \}$$

Resum

L'èxit del **Projecte Genoma Humà** (PGH) l'any 2000 va fer de la "medicina personalitzada" una realitat més propera. Els descobriments del PGH han simplificat les tècniques de seqüenciació de tal manera que actualment qualsevol persona pot aconseguir la seva seqüència d'ADN complerta.

La tecnologia de Read Mapping destaca en aquest tipus de tècniques i es caracteritza per manegar una gran quantitat de dades. **Hadoop**, el Framework d'*Apache* per aplicacions intensives de dades sota el paradigma **Map Reduce**, resulta un aliat perfecte per aquest tipus de tecnologia i ha sigut l'opció escollida per a realitzar aquest projecte. Durant tot el treball es realitza l'estudi, l'anàlisi i les experimentacions necessàries per aconseguir un Algoritme Genètic innovador que utilitzi tot el potencial de Hadoop.

Resumen

El éxito del **Proyecto Genoma Humano** (PGH) en el año 2.000 hizo de la "medicina personalizada" una relidad más cercana. Los descubrimientos del PGH han simplificado las técnicas de secuenciación de tal manera que actualmente cualquier persona puede conseguir su secuencia de ADN completa.

La tecnología de Read Mapping destaca en este tipo de técnicas y se caracteriza por manejar una gran cantidad de datos. **Hadoop**, el Framework de *Apache* para aplicaciones intensivas de datos bajo el paradigma **Map Reduce**, resulta un aliado perfecto para este tipo de tecnología y ha sido la opción escogida para realizar este proyecto. A lo largo del trabajo se realiza el estudio, el análisis y las experimentaciones necesarias para conseguir un Algoritmo Genómico novedoso que utilice todo el potencial de Hadoop.

Abstract

In the 2000th the **Human Genome Project** (PGH) was accomplished successfully and it made "personalized medicine" a closer reality. The PGH has simplified the sequencing techniques in a high way so nowadays anyone can get his full ADN sequence.

Read Mapping technology is one of most important sequencing techniques and it is characterized to work with lots of data. **Hadoop** is the Framework of *Apache* for data intensive applications under **Map Reduce** paradigm and it becomes a perfect tool for this kind of technology. For this reason it has been selected for this project. Along this entire project we will realize the study, the analysis and the experimentations to get a new Genetic Algorithm with all Hadoop potential.