Disseny i implementació d'un sistema de computació científica distribuïda utilitzant navegadors web i dispositius d'ús domèstic

David Romero Sánchez

Resum– Aquest article presenta el disseny i la implementació concreta d'un sistema de computació distribuïda heterogènia destinat a la recerca científica, utilitzant únicament navegadors web de dispositius d'ús quotidià. L'objectiu és oferir una solució eficient, segura, senzilla d'utilitzar i sense requerir cap tipus d'instal·lació o concessió de privilegis elevats. Es tracten també certs aspectes com l'optimització de l'ús de recursos i ample de banda del sistema central, utilitzant les tecnologies BitTorrent i WebRTC, la maximització de la compatibilitat amb els dispositius finals i l'aprofitament de les tecnologies Cloud per a suportar la infraestructura subjacent.

Paraules clau– computació científica, computació col·laborativa, computació distribuïda, navegadors web, Web Workers, WASM, BitTorrent, WebRTC

Abstract– This paper presents the design and concrete implementation of a heterogeneous distributed computing system intended for scientific research, using only web browsers of everyday devices. The goal is to offer an efficient, secure, easy-to-use solution without requiring any type of installation or granting of elevated privileges. Certain aspects are also discussed such as the optimization of the use of resources and bandwidth of the central system, using BitTorrent and WebRTC technologies, the maximization of compatibility with end devices and the use of Cloud technologies to support the underlying infrastructure.

Keywords– scientific computing, collaborative computing, distributed computing, web browsers, Web Workers, WASM, BitTorrent, WebRTC

1 Introducció

Aquest treball [1] neix com a resultat de la manca de recursos en molts àmbits de la ciència, i de les dificultats que això suposa de cara a realitzar investigacions "petites" o que no reben un finançament suficient en molts casos.

El projecte té com a objectiu crear un sistema de computació científica distribuïda heterogènia destinat a la recerca científica, utilitzant navegadors webs de dispositius d'ús quotidià. Gràcies a aquest sistema, el personal investigador adherit a la iniciativa podrà crear projectes dins la plataforma i carregar-hi tasques, les quals seran resoltes simultàniament de manera distribuïda per usuaris de tot el món que vulguin col·laborar aportant la capacitat de còmput sobrant dels seus dispositius domèstics (ordinadors, mòbils, tauletes...).

La solució resultant és una primera versió capaç de realitzar les funcions anteriorment mencionades amb uns estàndards mínims de seguretat i eficiència. Malgrat això, en cap cas es tracta d'un producte llest per entrar en un entorn de producció, ja que primer caldria realitzar un procés exhaustiu de revisió i millora, tant a nivell de funcionalitats com de seguretat i rendiment.

En un dels apartats posteriors d'aquest document, es tracta en més profunditat les futures accions que caldria realitzar, així com possibles millores que es podrien aplicar per fer que la plataforma complís els requisits necessaris per poder desenvolupar un paper en el món real.

• E-mail de contacte: david.romerosanc@autonoma.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Hing Fai Kevin Chow (Enginyeria de la Informació i de les Comunicacions)
• Curs 2022/23

1.1 Actors

Al sistema dissenyat trobem quatre actors principals, cadascun d'ells amb un rol i una capacitat d'actuació pròpia, i els quals són importants conèixer per tal d'entendre correctament les diferents parts de la solució.

1.1.1 Usuaris generals

Són persones que volen col·laborar amb la causa aportant la capacitat de còmput dels seus dispositius. Probablement no disposin de coneixements tecnològics elevats i les accions que han de realitzar per interactuar amb la plataforma són molt senzilles.

Accions que poden realitzar: Activar o desactivar la realització de tasques i consultar els projectes amb els que han col·laborat.

Requisits: Cap. Únicament realitzar un ús responsable de la plataforma i no malmetre-la voluntàriament amb intencions malicioses.

1.1.2 Investigadors/es

Personal docent o investigador que obté un benefici directe del sistema a través de l'obtenció de resultats per a les seves investigacions. Disposen de coneixements tecnològics suficients per crear el codi per a les tasques, així com compilarles i carregar-les a la plataforma.

Accions que poden realitzar: Crear nous projectes i tasques, modificar/eliminar projectes i tasques, aturar temporalment l'execució d'algunes tasques, i visualitzar l'estat i els resultats de les tasques.

Requisits: Han de ser membres d'alguna de les entitats adherides i es troben sota la seva responsabilitat.

1.1.3 Administradors/es

Personal de l'entitat que disposa de privilegis elevats en relació amb la gestió d'usuaris. En essència es tracta d'un perfil investigador que alhora disposa de capacitats d'administració dins la seva entitat.

Accions que poden realitzar: Totes les accions que pot realitzar el personal investigador, a més de poder crear i eliminar usuaris de la seva entitat, així com atorgar o retirar privilegis d'administrador a altres usuaris.

Requisits: Han de ser membres d'alguna de les entitats adherides i han d'actuar conforme amb la responsabilitat que se'ls hi ha atorgat.

1.1.4 Entitats

Universitats o institucions dedicades a la investigació que participen activament a la plataforma. Idealment aquest seria un projecte transversal en que diverses entitats serien les fundadores, i encarregades inicialment del finançament i l'administració. Posteriorment, altres entitats es podrien adherir, ja fos com a administradores juntament amb els membres originals o només com a usuàries.

Accions que poden realitzar: Tenen la capacitat i l'obligació de gestionar el seu personal que es troba a la plataforma, així com tots els projectes que estiguin realitzant.

Requisits: S'han d'haver adherit al projecte mitjançant un acord de col·laboració i acceptant unes condicions d'ús.

2 ESTAT DE L'ART

Al llarg del temps s'han portat a terme diversos projectes similars a aquest, on es creaven sistemes que permetien als usuaris contribuir amb la ciència aportant capacitat de còmput. Entre ells trobem "Zivis" [2], creat a l'estat espanyol l'any 2007 per contribuir en el desenvolupament del reactor nuclear de fusió "ITER", i "BOINC" de la Universitat de Berkeley [3], desenvolupat originalment com a part del projecte "SETI@home".

El projecte "BOINC" és el que es prendrà com a referència per a conèixer quin és l'estat actual de la computació distribuïda col·laborativa, ja que és l'únic que encara es troba en funcionament a dia d'avui.

2.1 Funcionament de BOINC

Per tal de participar en aquest projecte cal descarregar i instal·lar el seu client, tot i que addicionalment també recomanen instal·lar *VirtualBox* per qüestions de compatibilitat. A continuació, cal seleccionar el projecte en el que es vol participar i el client es connectarà amb els servidors corresponents. Després cal accedir o registrar-se amb un usuari i contrasenya, i si és la primera vegada que s'hi accedeix, també cal aportar informació addicional, així com unir-se a un grup en cas de desitjar-ho. Un cop realitzat aquest procés, es mostra una pàgina web amb informació relativa al compte creat on es poden editar les dades personals, modificar les preferències o veure els crèdits obtinguts a partir del còmput ofert al projecte seleccionat.

En quant a l'arquitectura, es tracta d'un sistema client-servidor en que el servidor entrega instruccions que el client haurà de computar, i reportarà els resultats de tornada. Els passos que es segueixen són els següents:

1. L'ordinador rep un conjunt de tasques del servidor de programació (*scheduling*) del projecte. Aquestes depenen de les capacitats de l'ordinador: per exemple, el servidor no donarà tasques que requereixin més memòria *RAM* de la que es disposa. Els projectes poden suportar diverses aplicacions i el servidor pot enviar tasques de qualsevol d'elles.

2. L'ordinador descarrega fitxers executables i d'entrada (*inputs*) des del servidor de dades del projecte. Si el projecte llança noves versions de les seves aplicacions, aquestes es descarreguen automàticament a l'ordinador.

3. L'ordinador executa els fitxers executables, produint fitxers de sortida.

4. L'ordinador carrega els fitxers de sortida al servidor de dades.

5. Més tard (fins a uns quants dies després depenent de les preferències) l'ordinador informa de les tasques completades al servidor de programació i s'obtenen tasques noves.

Aquest cicle es repeteix indefinidament.

2.2 Sistema de crèdit

Per tal d'incentivar la col·laboració dels voluntaris es porta a terme un sistema que ofereix crèdits als usuaris en funció del còmput realitzat. Per garantir que aquests crèdits s'entreguen de manera justa es segueix el següent esquema:

- Cada tasca es pot enviar a més d'un ordinador.
- Quan un ordinador informa d'un resultat, reclama una certa quantitat de crèdit, en funció del temps de *CPU* utilitzat.
- Quan s'han retornat almenys dos resultats, el servidor els compara. Si els resultats coincideixen, els usuaris reben el menor dels crèdits reclamats.

2.3 Punts febles

Entre les mancances detectades destaquen el fet que sigui necessari instal·lar software específic amb privilegis elevats, suposant això un possible problema de seguretat i privacitat, ja que cal confiar en l'entitat que ho desenvolupa, al mateix temps que un tercer podria arribar a manipular el software amb altres finalitats. També això suposa un impediment a la implantació massiva d'aquests sistemes, ja que moltes persones no estan disposades a realitzar l'esforç de seguir guies o tutorials per configurar els seus ordinadors per tal de col·laborar, mentre que sí ho farien si el sistema fos menys invasiu i més fàcil d'utilitzar. De la mateixa forma, les anteriors solucions estan dissenyades per sistemes operatius i arquitectures concretes i, mentre que per una banda això afavoreix el rendiment del sistema, també disminueix el públic que estarà disposat a utilitzar-ho.

3 Objectius

Aquest projecte té com a objectiu oferir una solució adaptada a la tecnologia actual i a com els usuaris en fan ús, millorant els punts febles de "BOINC". Això suposa oferir una solució *ready-to-go*, on l'usuari ha de fer un esforç mínim i es preveu que la seva col·laboració pot ser efímera, realitzant doncs tasques de curta durada i amb poques dependències, les respostes de les quals es comuniquen als investigadors en franges de temps petites.

Una possible resposta a aquesta situació és l'ús de navegadors web, degut a que es troben presents de manera predeterminada en pràcticament tots els dispositius informàtics que tenim al nostre abast. Això ofereix alhora diversos avantatges, com pot ser el fet de no haver d'instal·lar cap software addicional i poder col·laborar únicament accedint a una web. També permet actualitzar l'aplicació sense cap intervenció de l'usuari i utilitzar les últimes tecnologies disponibles segons aquestes van apareixent.

En definitiva, l'objectiu d'aquest projecte és crear una solució estable, eficient i segura que millori els següents aspectes:

- Execució de tasques independentment de la infraestructura subjacent.
- No requerir cap tipus d'instal·lació.
- No requerir privilegis elevats.
- Optimitzar l'enviament d'elements com tasques i fitxers de dades.

4 Metodologia

Aquest projecte s'ha desenvolupat seguint una metodologia iterativa àgil (*SCRUM* amb certes modificacions per adaptar-lo a la situació real d'aquest projecte), partint d'una fase de disseny general previ en que es van definir els conceptes més generals, així com l'arquitectura a implementar i les tecnologies que s'utilitzarien. La fase de desenvolupament de la solució es dividia en iteracions/*sprints*, sent cadascun d'aquests una part ben diferenciada del software a crear. Un exemple de *sprint* podria ser el disseny i creació del codi que es troba al client, i és l'encarregat d'executar les diferents tasques que rep garantint l'eficiència del sistema i evitant que l'ús de recursos superi els marges establerts per l'usuari o el sistema. Una vegada creat el codi, es realitzaven diverses proves d'ús per garantir el seu correcte funcionament i detectar possibles errors. Una vegada superada aquesta fase, es passava a desenvolupar un nou bloc i el cicle tornava a començar. Un cop es va finalitzar el desenvolupament, es va analitzar el codi per buscar possibles vulnerabilitats de seguretat, així com per intentar incrementar la seva eficiència.

5 Disseny del sistema

5.1 Lògica del sistema

5.1.1 Càrrega de fitxers de dades

El primer pas que cal realitzar per poder utilitzar la plataforma és carregar les dades que seran utilitzades durant la resolució de les tasques, en cas que aquestes les requereixin.

Aquesta acció l'ha de realitzar el personal investigador d'alguna de les entitats des del panell habilitat.

5.1.2 Creació de projectes i tasques

El següent pas, o el primer en cas que no existeixin dependències de dades, és crear un projecte i una tasca.

Un projecte és la representació d'una investigació que s'esta portant a terme i engloba un conjunt de tasques. A l'hora de crear-ne un de nou cal indicar un títol, un breu text descriptiu (una o dues línies), l'adreça web de la investigació i una imatge representativa.

Una tasca és, en essència, un problema que caldrà computar per tal d'obtenir un resultat. Aquests hauran de ser deterministes, per una mateixa entrada hauran de retornar un mateix valor, degut a que això és utilitzat en etapes posteriors per a la validació dels resultats.

Per a crear una tasca cal indicar un títol, el projecte al que pertany, un fitxer de codi, i les dependències de dades, en cas que en tingui.

5.1.3 Execució de tasques

Un cop ja existeixen tasques per resoldre a la plataforma, els usuaris generals entren en acció. Aquests han d'accedir a la web del projecte i prémer el botó per començar. Una vegada fet això, en segon pla es carregaran tasques i fitxers de dades, i es començaran a realitzar càlculs. Finalment, quan

l'usuari trobi un resultat, enviarà la resposta i demanarà una nova tasca. Aquest cicle es repetirà indefinidament fins que aturi l'execució, tanqui la finestra on s'estava executant o no existeixin noves tasques. En aquest últim cas, el client esperarà un minut i tornarà a sol·licitar-ne una de nova fins que la rebi o l'usuari indiqui la seva aturada.

5.1.4 Validació de resultats

La validació dels resultats enviats és realitzada pels propis usuaris. Per tal d'evitar que valors incorrectes es donin per vàlids, ja sigui degut a errors o accions malicioses, una mateixa tasca es distribueix a un nombre determinat de dispositius diferents, segons s'hagi definit a la configuració del sistema. Una vegada es rep aquest nombre de respostes iguals, la tasca es dona per finalitzada i el resultat s'accepta. En cas que cada usuari donés una resposta diferent, la tasca s'executaria indefinidament, tot i que també es podria establir un límit d'execucions per tal d'evitar que tingués lloc aquesta situació. Malgrat això, es tractaria d'un cas molt estrany que probablement fos degut a un atac extern o a un error a l'hora de crear el codi de la tasca.

5.1.5 Consulta de resultats

Una vegada el resultat ha sigut donat per vàlid, es pot consultar a través del panell de gestió dels investigadors, on es podrà descarregar per a utilitzar en altres entorns.

5.2 Arquitectura

El sistema disposa principalment d'una arquitectura client-servidor, on existeix un servidor que coordina l'entrega de les tasques i rep les respostes, i un client que és l'encarregat de rebre els problemes a resoldre, calcular-ne els resultats i retornar-los.

Addicionalment, el sistema també disposa d'un mòdul *Peer-To-Peer* (*P2P*) enfocat a la compartició dels fitxers de dades requerits per algunes tasques, tenint com a objectiu l'optimització de l'us d'ample de banda del servidor.

Quan s'afegeix una nova dependència de dades, el fitxer és carregat per l'usuari. A continuació, es crea un *Torrent* a partir d'aquest, però amb la particularitat que també s'indica un *web seed*. Això permet que quan ningú disposi encara del fitxer, aquest pugui ser descarregat directament des d'un servidor *HTTP*, però una vegada ja es trobi en circulació, únicament servirà per equilibrar la càrrega o garantir que el fitxer es trobi sempre disponible, ja que seran els propis usuaris els que realitzin gran part de l'esforç de compartir les dades.

5.2.1 Exemple

Per comprendre millor l'arquitectura descrita, es mostra a continuació un exemple de l'intercanvi d'informació entre els diferents elements:

1. L'usuari B sol·licita una nova tasca (recorregut groc).
2. El servidor fa una consulta a la base de dades per recuperar les tasques disponibles i selecciona la més indicada en funció del mètode (recorregut gris).
3. El servidor li envia la tasca a l'usuari, indicant-li que existeix una dependència de dades (fitxer 2) i on pot trobar el *Torrent* necessari per descarregar-la (recorregut taronja).
4. L'usuari B, utilitzant el *Torrent*, estableix una connexió amb l'usuari A i n'obté el fitxer de dades (recorregut blau).
5. Finalment, l'usuari B calcula el resultat de la tasca i l'envia de tornada al servidor (recorregut groc).
6. El servidor emmagatzema el resultat a la base de dades (recorregut gris) i queda a l'espera de noves peticions.

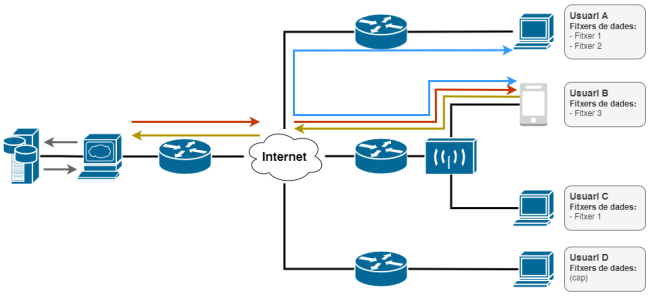

Fig. 1: Diagrama arquitectura

5.2.2 APIs

El *backend* està estructurat en forma d'*APIs*, concretament tres, cadascuna d'elles orientada a un rol d'usuari diferent. En primer lloc tenim l'*API* bàsica, la qual és oberta per a qualsevol usuari i l'únic requeriment que té és que cal enviar un identificador d'usuari amb cada petició. Entre les possibles peticions que es poden realitzar trobem l'obtenció d'informació sobre les contribucions realitzades, l'obtenció d'una nova tasca i l'enviament del resultat. A continuació, trobem l'*API* destinada al personal investigador, la qual requereix una identificació prèvia mitjançant un usuari i una contrasenya. Permet realitzar accions com crear i eliminar projectes i tasques, així com obtenir-ne els resultats. Finalment, tenim l'*API* del personal administrador. De la mateixa forma que l'anterior, aquesta requereix d'una autenticació prèvia, i alhora és necessari disposar del rol d'administrador. Permet crear i eliminar usuaris, així com atorgar-los permisos addicionals.

5.3 Llenguatges i tecnologies

5.3.1 Client

Per desenvolupar aquesta solució, es va optar per utilitzar *HTML5*, *CSS* i *JavaScript* a la pàgina principal, i addicionalment *Vue.js* a les pàgines internes (personal investigador i administrador).

Al mateix temps, la pàgina principal fa ús de diverses tecnologies presents per defecte als navegadors web com són els *Web Workers* [4], que permeten executar codi utilitzant fils en segon pla, *WebRTC* [5], que permet establir connexions *P2P* i compartir dades, funcions de les quals en fa ús *Web Torrent* [6], *IndexedDB* [7], que permet emmagatzemar de manera persistent dades de qualsevol tipus en forma

de taules sense limitacions d'espai (més enllà de les limitacions del propi disc), i *Web Storage - Local Storage* [8], utilitzat per guardar cadenes de text relativament petites de manera persistent.

5.3.2 Servidor

Al costat del servidor es va decidir utilitzar *Node.js* degut principalment al seu rendiment i a la familiaritat, ja que així els temps de desenvolupament serien molt menors.

També es va utilitzar el *framework* web *Express*, el qual permet gestionar de manera senzilla les peticions *HTTP* rebudes, així com realitzar funcions de *middleware* en qualsevol punt de la cadena de peticions.

Addicionalment, es va utilitzar la llibreria *express-session* per a la gestió de les sessions dels usuaris, *bcrypt* per calcular el *hash* de les contrasenyes, *create-torrent* per crear els fitxers *Torrent* de les dependències de dades, *multer* per gestionar els fitxers enviats pels usuaris a través dels panells de control, *nodemailer* per enviar els correus electrònics des de la propia aplicació, i les llibreries pròpies de *Google* que permeten la connexió amb el núvol.

5.4 Client

A continuació es mostren els diversos fitxers en que es divideix el client, així com una breu explicació del seu contingut i la seva funcionalitat.

5.4.1 main.js

La seva funció és la de servir com a punt d'entrada a les diferents funcionalitats del client, alhora que les coordina amb les accions de l'usuari i modifica la interfície que utilitza l'usuari. Comprova si el dispositiu compleix els requisits mínims per executar l'aplicació i en cas de no ser així, redirigeix a l'usuari a una pàgina alternativa. En cas que les verificacions siguin satisfactòries, inicialitza els diferents subsistemes (còmput, *P2P* i base de dades local).

5.4.2 compute.js

Conté la classe *Compute* i la seva principal funció consisteix en obtenir noves tasques, executar-les utilitzant *Web Workers*, i enviar els resultats obtinguts de tornada al servidor. Quan aquesta classe s'inicialitza, alhora arrenca el subsistema *P2P* i la base de dades local.

5.4.3 p2p.js

Conté la classe *P2P* i és l'encarregada de gestionar la connexió i l'intercanvi de dades amb altres usuaris de la plataforma. Permet descarregar dades a partir de fitxers *Torrent* i compartir els fitxers dels que ja disposa. Quan aquesta classe s'inicialitza, arrenca alhora la base de dades local.

5.4.4 database.js

Conté la classe *Database* i té com a funció mantenir i operar la base de dades local utilitzant l'*API* d'*IndexedDB*. Permet emmagatzemar localment i de manera persistent les dependències de dades, evitant així que s'hagin de tornar a descarregar quan una nova tasca les torna a requerir.

5.4.5 wasm-worker.js

Conté el codi necessari per a executar els *Web Worker*. Està preparat per rebre el codi a executar i les dades necessàries a través de *postMessage()*, i una vegada ha realitzat el còmput i disposa d'un resultat, el retorna utilitzant el mateix mètode.

5.4.6 config.js

Fitxer de configuració que disposa de variables que es poden ajustar segons es consideri per modificar el comportament del client.

5.5 Servidor

A continuació, es mostren els diversos fitxers en que es divideix el servidor, així com una breu explicació del seu contingut i la seva funcionalitat.

5.5.1 index.js

Punt d'entrada de *Node.js*. Defineix les sessions d'*express-session* i agrupa els diversos encaminadors que s'utilitzaran (*api*, *entities* i *admin*).

5.5.2 api.js

Conté les rutes de l'*API* que utilitzaran els usuaris generals.

5.5.3 entities.js

Conté les rutes de l'*API* que utilitzarà el personal investigador.

5.5.4 admin.js

Conté les rutes de l'*API* que utilitzarà el personal administrador.

5.5.5 database.js

Conté la classe "Database", la qual permet realitzar totes les accions que involucren a la base de dades, com per exemple, verificar si les dades d'inici de sessió d'un usuari són correctes o modificar les dades d'una tasca.

5.5.6 email.js

Conté la classe "Email" i permet enviar correus electrònics directament des de l'aplicació. En aquest cas, s'utilitza per donar la benvinguda als nous usuaris i proporcionar-los una contrasenya temporal per iniciar sessió la primera vegada, alhora que permet enviar noves contrasenyes en cas que s'hagin restablert.

5.5.7 options.js

Conjunt de variables que permeten ajustar el comportament de la solució. Actualment trobem opcions per configurar aspectes com el nombre de respostes iguals que cal rebre per donar una solució per vàlida o el mètode utilitzat per a la distribució de tasques.

5.6 Compatibilitat

Un dels objectius més importants a assolir era que el sistema fos capaç d'executar-se en el major nombre possible de dispositius, independentment de la arquitectura dels mateixos. És per això que es van seleccionar tecnologies web que, a banda d'oferir les capacitats tècniques necessàries, es trobessin consolidades i fossin compatibles amb la gran majoria de navegadors.

Per a verificar això, es va utilitzar la informació continguda a la web del projecte "Can I use" [9]. Les taules de versions utilitzades es poden consultar a l'apèndix A.1.

5.7 Seguretat

5.7.1 Limitació dels temps d'execució

A través del fitxer de configuració és possible establir una limitació en els temps d'execució de les tasques en els dispositius dels usuaris. D'aquesta forma, s'evita que un codi defectuós o maliciós es pugui executar de manera indefinida, afectant al rendiment del dispositiu.

5.7.2 Identificació d'usuaris generals

Per tal de portar un control sobre els usuaris que utilitzen la plataforma, alhora que es respecta la seva privacitat, es va optar per implementar un sistema d'identificadors que calgués utilitzar per poder interactuar amb el sistema. Això permet, per exemple, vincular les execucions de múltiples tasques amb un mateix usuari, i d'aquesta forma saber quines han estat les seves aportacions. Malgrat això, en cap cas és un mètode de protecció infal·lible, ja que és extremadament fàcil modificar aquest identificador.

5.7.3 Autenticació d'usuaris privilegiats

Degut a que certes accions es troben restringides a un petit col·lectiu, és imprescindible la creació d'un sistema d'autenticació. En aquest cas, s'ha optat per utilitzar una versió molt simple, en que es requereix un usuari (direcció de correu electrònic) i una contrasenya, la qual es troba emmagatzemada a la base de dades del servidor utilitzant una funció de resum (*hash*).

6 DETALLS DE LA IMPLEMENTACIÓ

6.1 Distribució de tasques

Malgrat no estar contemplat en el disseny inicial, durant el desenvolupament de la solució es va decidir adaptar el codi encarregat de distribuir les tasques per acceptar diversos mètodes. Això possibilita que en funció de l'ús de la plataforma o del rendiment observat, sigui possible escollir un mètode o un altre únicament modificant una variable.

Actualment existeixen dos mètodes. El primer s'anomena "simpleTaskDistribution", i com el seu nombre indica, és el més senzill. Un cop rep el llistat de tasques pendents d'executar, escull la primera, sempre i quan aquesta no hagués estat ja executada pel mateix usuari. El segon s'anomena "fairTaskDistribution", i la seva principal diferència és que en aquest cas ordena les tasques en funció del nombre d'execucions, prioritzant les tasques que menys cops han estat executades.

6.2 Llenguatges suportats per a les tasques

En una primera versió, es va desenvolupar una solució que únicament era capaç d'executar codi *JavaScript* utilitzant *Web Workers*. Malgrat ser funcional, es va arribar a la conclusió que això no era ideal, ja que en la realització de tasques científiques o de computació intensiva, el rendiment i l'eficiència són claus. Al mateix temps, és molt probable que ja es disposi de fragments de codi creats prèviament en llenguatges com *C* i *Rust*, i en cas de suportar únicament *JavaScript*, caldria refer-los. És per això que finalment es va optar per crear una solució que executés tasques utilitzant *WebAssembly* (*WASM*), que pot ser compilat a partir de codi existent en *C*, *Rust*, *Go* i pràcticament qualsevol llenguatge que utilitzi *LLVM* per a la seva compilació. Malgrat això, es podria crear una nova versió de la solució que suportes ambdós mètodes, degut a que en alguns casos podria ser preferible utilitzar *JavaScript* degut a la simplicitat de creació i execució, i a la manca d'un pas addicional de compilació.

6.3 Compilació WASM

Degut a les dificultats derivades de la falta de maduresa de *WASM* a nivell d'interfícies de dades, en aquesta primera versió únicament s'ha considerat la compilació a partir de codi escrit en *C*. Per a aquest objectiu s'ha pres com a referència el compilador *Emscripten* [10], degut a que és un dels més populars a l'actualitat. Tot i així, existeixen altres alternatives, com per exemple *WasmFiddle* [11], que permet compilar codi en *C* directament des del navegador i sense haver d'instalar dependències.

6.3.1 Emscripten

És un compilador basat en *LLVM/Clang* que compila el codi font *C* i *C++* a *WebAssembly*, principalment per a l'execució en navegadors web.

Emscripten permet que les aplicacions i biblioteques escrites en *C* o *C++* es compilin anticipadament i s'executin de manera eficient en navegadors web, normalment a velocitats comparables o superiors a les del *JavaScript* interpretat o compilat dinàmicament. Fins i tot emula un sistema operatiu *POSIX* sencer, permetent als programadors utilitzar funcions de la biblioteca estàndard *C* (*libc*).

6.3.2 WasmFiddle

És una eina d'edició de codi en línia que permet escriure codi *C* o *C++* i convertir-lo a *WebAssembly Text Format* (*WAT*), compilar-lo a *WASM* o interactuar-hi directament mitjançant *JavaScript*. Els editors *C/C++* i *JavaScript* ofereixen una funcionalitat mínima i no estan pensats per ser utilitzats com a entorn de desenvolupament principal, però ofereixen una alternativa senzilla per a poder realitzar proves.

6.4 Dependències de dades

Quan s'utilitza *Emscripten*, existeix l'opció de compilar el codi juntament amb els fitxers de dades necessaris per a l'execució, utilitzant l'opció --preload-file". Això fa que les dades es trobin incloses dins del sistema de fitxers virtual

creat per *Emscripten*, i puguin ser accedides de manera directa utilitzant la llibreria "stdio.h".

Malgrat ser molt convenient a nivell d'execució, això fa que tots els codis generats hagin d'estar lligats a les dades que requereixen, provocant el malbaratament d'ample de banda en cas que un mateix conjunt de dades sigui requerit per múltiples tasques diferents, ja que aquest s'hauria d'enviar repetidament amb cada execució.

Per tal de solucionar aquesta situació, es va optar per separar completament les tasques de les dependències de dades. Per una banda, es gestionaria l'enviament de tasques, i per una altra les dependències.

La dificultat en aquest cas consistia en que no existeix una forma directa i senzilla d'intercanviar estructures complexes de dades amb un mòdul *WASM*, ja que per defecte, actualment només es suporten quatre tipus de dades: sencers de 32 i 64 bits, i valors de punt flotant de 32 i 64 bits. També existeixen els *Reference Types*, com per exemple *externref* que permeten emmagatzemar qualsevol valor de *JavaScript*, com cadenes de text i objectes, però aquests són opacs des del punt de vista de *Web Assembly*, ja que el mòdul *WASM* no pot accedir ni manipular aquests valors i només pot realitzar accions com crides a funcions. És per això que no serveix per resoldre aquesta situació.

Finalment, es va escollir prendre una aproximació basada en la memòria que utilitza el mòdul *WASM*, ja que aquesta és accessible des de *JavaScript*. Quan s'executa una tasca que requereix dades externes, primer es recupera el fitxer (independentment del tipus de dades i l'extensió) utilitzant el subsistema *P2P* i la base de dades local. Una vegada es disposa d'aquest, el contingut es converteix en una cadena de bytes i es calcula la seva mida. A continuació, es creen noves pàgines de memòria al mòdul *WASM*, en funció de la mida de les dades, i es copien allà. Finalment, s'executa la funció "compute" però, a diferència d'altres casos, amb dos paràmetres. El primer és el desplaçament que indica a quina part de la memòria comencen les dades proporcionades, en essència un apuntador, i el segon paràmetre indica la mida de les dades. Disposant d'aquesta informació, l'únic que cal fer llavors és processar les cadenes de bytes en funció del tipus de dades que fossin inicialment, per a continuació, executar la tasca en sí mateixa.

```
const dataArray = new Uint8Array(fileBuffer);
const dataSize = dataArray.length;
const currentSize = exports.memory.buffer.
    byteLength;
const neededPages = Math.ceil(dataSize /
    65536); // 64KB pages
exports.memory.grow(neededPages);
const offset = currentSize;
const dataView = new Uint8Array(exports.
    memory.buffer, offset, dataSize);
dataView.set(dataArray);
```

Fig. 2: Fragment de codi que copia les dependències de dades al mòdul *WASM*

6.5 Retorn de resultats

El retorn de resultats segueix sent un problema degut a la mateixa situació comentada a l'apartat anterior, la impossibilitat de compartir tipus de dades complexes amb un mòdul *WASM* de manera senzilla.

Per aconseguir el retorn dels resultats, independentment del seu tipus, es segueix la mateixa estratègia. Totes les tasques que es volen executar al sistema cal que disposin de dues funcions. La primera és la funció "compute", la qual conté tota la lògica de la tasca i realitza crides a funcions secundàries si així es requereix. Aquesta haurà de retornar un apuntador al resultat. També caldrà que la tasca disposi d'una funció anomenada "resultSize" que retorni un valor sencer, el qual indicarà la mida del resultat. D'aquesta forma, és molt senzill recuperar els valors directament des de la memòria, independentment de la seva mida o format.

Novament, una vegada extrets els valors, caldrà processar-los per obtenir informació llegible, aplicant les transformacions necessàries per disposar finalment del format de dades desitjat.

```
const resultPtr = exports.compute(offset,
    dataSize);
const resultSize = exports.resultSize(
    resultPtr);
resultArray = new Uint8Array(
  exports.memory.buffer,
  resultPtr,
  resultSize
);
```

Fig. 3: Fragment de codi que executa la funció principal i recupera el resultat

7 DESPLEGAMENT AL NÚVOL

Per tal de garantir la disponibilitat de la plataforma independentment del nombre d'usuaris i la càrrega del sistema, es va adaptar el codi per poder desplegar l'aplicació al núvol de manera nativa. En aquest cas, es va optar per utilitzar el núvol de *Google*, conegut com a *Google Cloud Platform (GCP)*, principalment per la familiaritat amb el mateix i degut a que els productes oferts encaixaven amb les necessitats.

7.1 App Engine

És una solució de plataforma com a servei (*PaaS*) de computació al núvol per crear aplicacions escalables. Permet als desenvolupadors crear i allotjar aplicacions web sense haver d'aprovisionar servidors ni preocupar-se per les actualitzacions del sistema, ja que es tracta d'una plataforma completament administrada sense servidor (*serverless*).

S'ha utilitzat per allotjar l'aplicació desenvolupada en *Node.js* que actua com a servidor central. És l'encarregada d'entregar les pàgines quan aquestes són sol·licitades, respondre a les crides de l'*API*, actuar com a intermediària amb *Cloud Storage* i *Firestore*, i en general realitzar totes les tasques pròpies del servidor.

7.2 Firestore - Datastore

És una base de dades *NoSQL* orientada a documents, escalable i amb disponibilitat global. Permet crear jerarquies per emmagatzemar dades relacionades i recuperar-les

mitjançant consultes expressives de manera senzilla. Totes les consultes s'escalen amb la mida del conjunt de resultats (i no amb el del conjunt de dades), de manera que l'aplicació pot créixer amb molta facilitat.

S'ha utilitzat com a base de dades per emmagatzemar tota la informació relativa a la plataforma, com per exemple, dades d'usuaris, projectes, tasques, execucions, dependències de dades...

7.3 Cloud Storage

És un servei per emmagatzemar objectes al núvol, garantint la seva disponibilitat. Un objecte és en essència un fitxer de dades de qualsevol format. Aquests s'emmagatzemen en contenidors anomenats *buckets*. Tots els *buckets* estan associats a un projecte que, alhora, es pot associar a una organització.

S'ha utilitzat per emmagatzemar tots els fitxers de dades utilitzats per la plataforma, des de les imatges identificatives dels projectes, fins al codi de les tasques i els fitxers *Torrent* que permeten descarregar les dependències de dades.

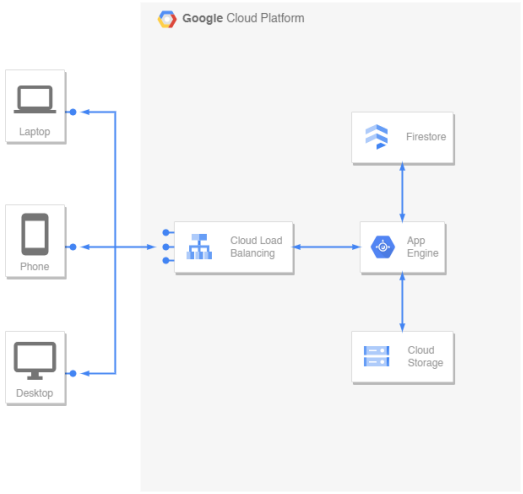

Fig. 4: Arquitectura Google Cloud

8 Resultats

El resultat final obtingut és una plataforma completament funcional i capaç d'operar de manera autònoma, ajustant els recursos per adaptar-se al tràfic i a les necessitats.

Per tal de conèixer si la solució resultant assoleix uns nivells mínims de rendiment, s'han portat a terme diversos tests.

8.1 Rendiment web

Durant el desenvolupament de la plataforma, es va prioritzar el rendiment, especialment de la pàgina principal. És per això, que es va prescindir de l'ús de qualsevol *framework* o codi extern que no fos estrictament imprescindible, reduint així els temps de càrrega.

Per tal de verificar que aquest objectiu s'havia assolit, es va realitzar el conjunt de proves de *Lighthouse* [12], que permet conèixer de manera ràpida quin és l'estat de la web i quines són les millores que caldria aplicar.

Els resultats indicaven que la web complia els estàndards a nivell d'accessibilitat, bones pràctiques i *SEO*, alhora que oferia un molt bon rendiment general. Algunes de les recomanacions que es proporcionaven per millorar encara més el rendiment de la pàgina eren:

- Establir explícitament l'alçada i amplada de totes les imatges.
- Ajornar la càrrega de les imatges que no apareguessin just al carregar la pàgina.
- Minimitzar la càrrega del *thread* principal.
- Eliminar els recursos que bloquegen la renderització de la pàgina.

Malgrat alguns d'aquests punts no es podien millorar perquè els recursos als que feien referència eren estrictament necessaris, es van aplicar la resta de millores, fent que els resultats obtinguts fossin encara millors.

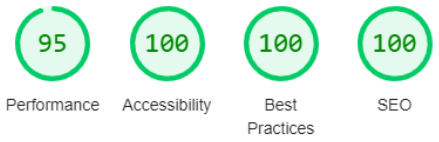

Fig. 5: Resultats inicials tests Lighthouse

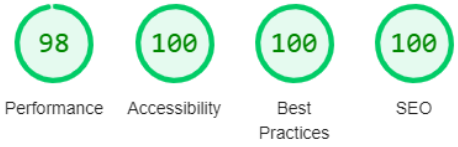

Fig. 6: Resultats finals tests Lighthouse

8.2 Rendiment P2P

També es va quantificar el rendiment del subsistema *P2P*, però degut a que depèn de molts factors externs com els rastrejadors, és important tenir en compte que aquests valors poden diferir molt.

En primer lloc, es va intentar carregar al sistema un fitxer relativament pesat (82,9 MB), el qual va retornar un error 413 (*Request Entity Too Large*) i va permetre descobrir que les peticions entrants a *Google App Engine* tenen una limitació de 32 MB. Això fa que en futures versions calgui modificar el sistema de càrrega per permetre que els usuaris interactuïn directament amb *Google Cloud Storage*, evitant així aquesta limitació. Per tal d'evitar això temporalment, es va carregar un fitxer de 22 MB, el qual va ser posteriorment descarregat per un usuari que va resoldre una tasca associada. El temps de descarrega va ser d'uns 4 minuts. A continuació, un altre usuari va resoldre la mateixa tasca, activant així el procés d'intercanvi del fitxer utilitzant *P2P*. En aquest cas, el temps de descarrega va ser de només 12 segons. Aquesta diferència tan notable probablement sigui deguda a que en el segon cas, ambdós dispositius es trobaven dins la mateixa xarxa local, reduint així els salts que ha de fer i accelerant el procés. Al mateix temps, com en

el primer cas cap usuari disposava del fitxer, aquest es descarregava des de *Google Cloud Storage* com un *web seed*. Degut a la forma en que està implementat, es possible que això reduís en gran mesura les velocitats, i una possible millora per futures versions és que en el cas que pocs o cap usuari disposés del fitxer, en comptes de recórrer al subsistema *P2P*, aquest s'obtingués directament via *HTTP*.

9 ÚS EN UN ENTORN REAL

Per tal de verificar si la solució desenvolupada encaixaria amb les necessitats actuals dels equips de recerca, es va intentar contactar amb diversos/es investigadors/es per conèixer el seu punt de vista.

Les primeres preguntes estaven enfocades a conèixer com és el procés de sol·licitud de recursos computacionals i quin és el temps d'espera de mitjana. La resposta obtinguda va ser que habitualment les sol·licituds es fan directament a col·laboradors o s'utilitzen recursos gratuïts. És per això, que en aquests casos no té gaire sentit considerar la variable del temps, ja que tot plegat es resol de manera interna i no cal recórrer a entitats externes.

El segon grup de preguntes anava dirigit a les càrregues de treball, així com els equips on s'executen. En aquest sentit, cal tenir en compte que la complexitat de les tasques a executar pot variar molt, i els recursos necessaris per a cada cas poden ser molt diferents. Per exemple, les tasques més habituals solen ser l'anàlisi de grans volums de dades utilitzant *R*. Quan aquest volum entra dins d'uns límits assumibles, es solen utilitzar equips propis del grup de recerca, però si la feina és més exigent, sovint s'utilitzen equips d'investigadors col·laboradors, que solen formar part d'entitats externes com ara centres d'investigació o universitats.

Finalment, les últimes preguntes eren d'opinió personal, i van resultar ser les més interessant. Una d'elles preguntava si es considerava que els recursos computacionals disponibles són suficients per satisfer les necessitats de la majoria dels grups de recerca. La resposta va ser que el principal problema en aquest sentit és que es coneixen poc quins recursos existeixen i hi ha poca formació per poder-los utilitzar. És per això que habitualment s'ha de recórrer a col·laboradors més experts en aquests temes o contractar personal especialitzat.

Això permet visualitzar quin pot ser el pla de ruta del projecte, per tal d'adreçar els diversos problemes existents en l'actualitat. Entre d'altres aspectes que pot aportar, trobem la simplificació de l'accés als recursos, ja que a través d'un panell d'administració molt simple es poden crear noves tasques ràpidament, així com visualitzar-ne o descarregar els resultats. Al mateix temps, les capacitats multillenguatge de la plataforma permeten que els investigadors reutilitzin el codi que ja tenen o en creïn de nou, però utilitzant les eines que ja coneixen.

10 FUTURES MILLORES

10.1 Funcionalitats

10.1.1 Simplificació de les dependències de dades

El sistema actual de dependències pot arribar a ser bastant dens d'utilitzar, tenint en compte que cal accedir de manera directa a la memòria per poder obtenir les dades i aplicar les transformacions necessàries. Això podria ser millorat incorporant funcions predefinides que permetessin obtenir i transformar la informació automàticament en funció del seu tipus. Això es pot fer creant funcions d'entorn que s'entreguen a l'hora de fer la instanciació del codi *WASM*. L'únic inconvenient és que aquestes serien funcions específiques de la plataforma i únicament estarien disponibles quan el codi s'executés a la mateixa.

Una altra solució, la que seria més òptima, és esperar a que *Web Assembly* suporti de manera nativa l'intercanvi de tipus de dades com cadenes de text entre mòduls de *WASM*.

10.1.2 Neteja de la base de dades

Per tal de garantir un rendiment òptim del sistema, es podrien establir certs mecanismes que permetessin netejar la base de dades local, eliminant els fitxers que fa molt temps que no s'utilitzen. D'aquesta forma, tot i que no impactaria de forma directa en els temps d'execució, s'optimitzaria l'ús dels recursos dels dispositius dels usuaris.

10.1.3 Pati de proves

Una possible funcionalitat que facilitaria enormement la feina al personal que crea les tasques és un "pati de proves", el qual permetés provar localment l'execució d'una tasca i les seves dependències. D'aquesta forma, es podria garantir que el comportament fos l'esperat i no existissin errors que calgués reparar.

10.2 Seguretat

10.2.1 Limitació d'ús per IP

Actualment, els usuaris generals només són identificats per una cadena de text que han d'enviar amb cada petició/resposta, però no s'emmagatzema cap altra peça d'informació relacionada. L'emmagatzematge de la direcció IP juntament amb l'identificador, tot i que podria tenir certes implicacions a nivell de protecció de dades, permetria establir un control sobre les accions dels usuaris i prohibirne l'accés als que no respectessin les condicions d'ús. Per exemple, es podria impedir l'accés a la plataforma a un usuari durant un temps predefinit en cas que enviï moltes respostes que difereixen del que afirmen la resta d'usuaris.

10.2.2 Verificació de les tasques carregades

Realització d'un primer anàlisi estàtic automàtic del codi de les tasques carregades a la plataforma, per tal de garantir que no realitzi cap tipus d'acció maliciosa i no contingui, per exemple, bucles infinits, que puguin perjudicar el rendiment dels dispositius dels usuaris.

10.2.3 Seguiment de les tasques enviades

Tot i que aquesta problemàtica hauria de quedar resolta en gran mesura amb la limitació per IP, una millora recomanable seria aplicar un seguiment de les tasques enviades als usuaris per ser resoltes. D'aquesta forma, únicament els usuaris que han rebut una tasca concreta tindran la capacitat de presentar una resposta.

Actualment, un usuari pot enviar una resposta per a qualsevol tasca sempre que conegui el seu identificador, el qual no és públic i per tant tampoc hauria de tenir a la seva disposició. Malgrat això, si s'apliqués un seguiment de les tasques enviades es podria reduir encara més el nombre de respostes incorrectes rebudes, ja sigui per errors o per accions malicioses.

10.3 Incentius

Per tal d'aconseguir que un major públic accedís a la web i col·laborés amb la causa, caldria trobar nous mètodes d'incentiu que satisfessin als usuaris. Aquests podrien ser proposats per les pròpies entitats que formessin part del projecte, i podrien estar relacionats d'alguna forma amb les investigacions que s'estiguessin portant a terme.

10.4 Rendiment

10.4.1 Primeres descarregues

Tal i com s'ha observat en l'apartat de resultats, seria recomanable modificar el sistema *P2P* per tal que les primeres descarregues que es realitzin d'un fitxer, utilitzin el protocol *HTTP* directament amb el servidor de *Google Cloud Storage*. Això permetria aprofitar l'ample de banda que ofereix i s'aconseguiria que les descarregues fossin molt més ràpides.

11 Conclusions

En aquest treball, s'ha dissenyat i implementat un sistema de computació científica distribuïda heterogeni basat en navegadors webs. Aquest permet que persones corrents puguin col·laborar de manera senzilla amb diverses investigacions científiques, aportant la capacitat de còmput dels seus dispositius quotidians que no utilitzen.

Per a aconseguir això, s'han creat tres aplicacions web que permeten que cada usuari, en funció del seu rol, realitzi un conjunt d'accions. Des de resoldre problemes computacionals a crear noves tasques i gestionar usuaris.

Per tal de portar aquestes capacitats a un gran nombre de dispositius sense requerir cap tipus d'instal·lació, s'han utilitzat diverses tecnologies web actuals suportades per la gran majoria de navegadors web. Entre elles destaquen *WASM*, per a l'execució de codi compilat a partir de llenguatges com *C/C++*, *Rust* i *Go*, amb un rendiment similar al que s'obtindria nativament, *Web Torrent*, que està creat a partir de *WebRTC* i el protocol *BitTorrent*, i que permet l'intercanvi de fitxers de manera directa entre els usuaris (*P2P*), i altres *APIs* que es troben per defecte als navegadors web com *IndexedDB*, per emmagatzemar dades estructurades, i *Web Workers*, per executar codi de manera aïllada i sense afectar al fil principal d'execució.

Tot i no tractar-se d'una solució final, aquest projecte assenta les bases per a la creació d'un sistema de computació científica unificat, el qual permeti executar tasques independentment de l'arquitectura i de les tecnologies de software utilitzades.

Agraïments

El meu més sincer agraïment pel Kevin Chow, qui ha tutoritzat aquest treball i m'ha acompanyat durant tot el procés, aconsellant-me i resolent tots els dubtes que pogués tenir. També a la Roser Masgrau, professora i investigadora de l'Institut de Neurociències, per respondre a les meves preguntes sobre el món de la recerca. I finalment, agrair enormement a la meva família, amics i companys que m'han recolzat durant tota la meva vida universitària i han estat allà sempre que ho he necessitat.

Referències

[1] ≪darosa01/hive-compute: A distributed scientific computing system using web browsers and everyday devices.≫ https://github.com/darosa01/hive-compute

[2] ≪Portada Zivis - Ayuntamiento de Zaragoza≫ https://www.zaragoza.es/contenidos/conocimiento/zivis/DipticoZivis.pdf

[3] Anderson, D.P. BOINC: A Platform for Volunteer Computing. J Grid Computing 18, 99–122 (2020). https://doi.org/10.1007/s10723-019-09497-9

[4] ≪Web Workers API - Web APIs — MDN≫, 26 febrer 2023. https://developer.mozilla.org/es/docs/Web/API/Web_Workers_API

[5] ≪WebRTC API - Web APIs — MDN≫, 5 juny 2023. https://developer.mozilla.org/es/docs/Web/API/WebRTC_API

[6] ≪webtorrent/webtorrent: Streaming torrent client for the web≫, GitHub. https://github.com/webtorrent/webtorrent

[7] ≪IndexedDB API - Web APIs — MDN≫, 21 març 2023. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

[8] ≪Web Storage API - Web APIs — MDN≫, 25 maig 2023. https://developer.mozilla.org/es/docs/Web/API/Web_Storage_API

[9] ≪Can I use... Support tables for HTML5, CSS3, etc≫. https://caniuse.com/

[10] ≪Main — Emscripten 3.1.42-git (dev) documentation≫. https://emscripten.org/

[11] ≪wasdk/WasmFiddle: WebAssembly Fiddle≫, GitHub. https://github.com/wasdk/WasmFiddle

[12] ≪Lighthouse overview≫, Chrome Developers, 27 setembre 2016. https://developer.chrome.com/docs/lighthouse/overview/

APÈNDIX

A.1 Taules de compatibilitat

A.1.1 Web Storage - Local Storage

Aquesta tecnologia es troba actualment suportada per un 97,44% dels navegadors en ús.

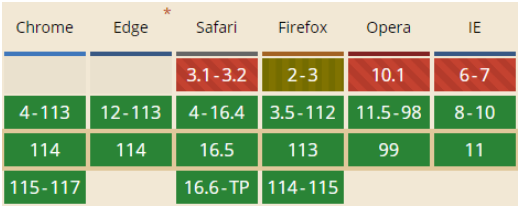

Fig. 7: Taula de compatibilitat - Web Storage

A.1.2 Web Workers

Aquesta tecnologia es troba actualment suportada per un 97,2% dels navegadors en ús.

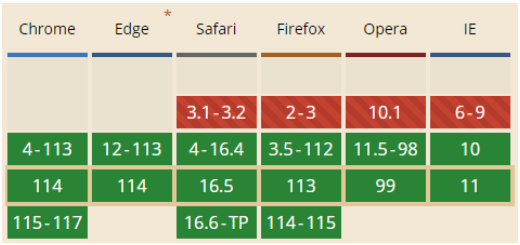

Fig. 8: Taula de compatibilitat - Web Workers

A.1.3 IndexedDB

Aquesta tecnologia es troba actualment suportada per un 97,16% dels navegadors en ús.

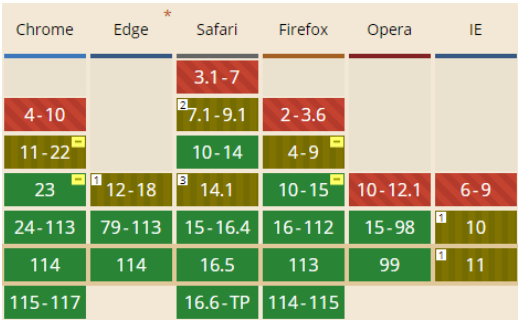

Fig. 9: Taula de compatibilitat - IndexedDB

A.1.4 WebRTC

Aquesta tecnologia es troba actualment suportada per un 96,36% dels navegadors en ús.

Anteriorment, Edge no suportava les funcions de RTC-DataChannel degut a que el sistema operatiu Windows no suportava el protocol *Datagram Transport Layer Security* (DTLS) de manera nativa, el qual era necessari per establir canals de dades utilitzant el protocol *Stream Control Transmission Protocol* (SCTP). Això va canviar amb l'ús de *Chromium* com a motor del navegador.

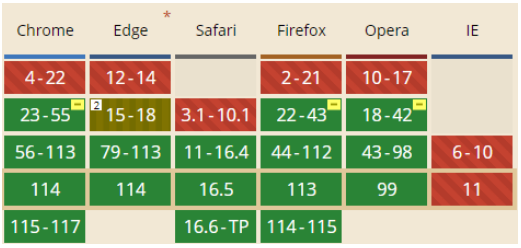

Fig. 10: Taula de compatibilitat - WebRTC