Rövid futásidejű alkalmazások kezelése cloud rendszerekben

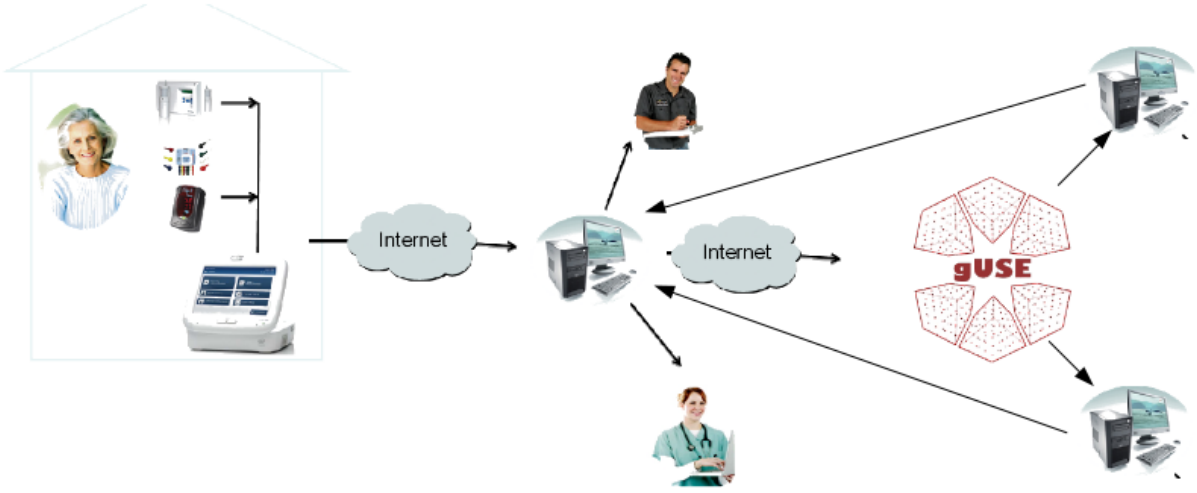
ABSZTRAKT

Napjainkban a számításigényes kutatások a klaszterekről, a gridekről és akár még a szuperszámítógépekről is egyre inkább a cloud rendszerek felé vándorolnak. Ez azonban egy teljesen más környezet, mások a "szabályok", amik nem mindig segítik a munkánkat. Amikor egy korábban griden működő számítási alkalmazásunkat cloud környezetbe ültetünk át számtalan akadályt kell legyőznünk. Az sem meglepő ha kezdetben egyáltalán nem működik az alkalmazásunk, vagy jóval lassabban mint amit megszoktunk korábban. A kérés magától adódó: hogyan lehet ezen javítani és technológiában rejlő egyértelmű előnyöket a szolgálatunkba állítani?

Mit kell nekünk tenni, és milyen rendszerek nyújtanak számunkra hatékony segítséget abban, hogy saját alkalmazásaink végrehajtásában is előrelépést érhessünk el cloud környezetben? Az előadás során meglévő parametrikus alkalmazást fogunk átültetni gridről cloud-ra. Végignézzük a lehetséges buktatókat, és teljesítmény növelési lehetőségeket és kiszámoljuk mennyit nyertünk vagy vesztettünk a migráción.

Bevezetés

Az Óbudai Egyetem Neumann János Informatikai Karán működő Biotech labor [1] kutatásaiban részt vevő felhasználók egészségügyi paramétereit (ECG, Vérnyomás, Vércukor stb) mobil egészségügyi eszközökkel mérik. A mért értékeket a mobiltelefon segítségével eljutnak egy központi adattárolóban, ahonnét webes felületen az arra jogosult szakemberek akár historikusan is visszanézhetik ezeket. A rögzített adatokat több algoritmus folyamatosan feldolgozza, és a számított értékeket is visszatölti az adatbázisba. Az adatok feldolgozása az MTA SZTAKI Párhuzamos és Elosztott Rendszerek Laboratoriumának WS-PGRADE/gUSE/DCI-Bridge [2] rendszereinek segítségével történik.

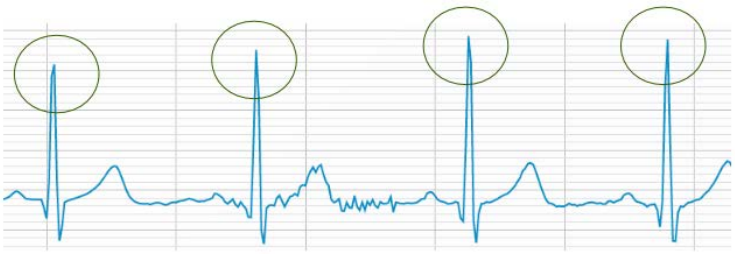


1. ábra: A rendszer felépítése

A használt algoritmusok működése és felépítése nagyon eltérő. Egyes algoritmusok több program végrehajtását igénylik, mások csak egy alkalmazásból állnak. Vannak olyan algoritmusok, amelyek szinte folyamatosan futnak, mások csak időnként. Vannak olyan algoritmusok, amelyek a felhasználó aktuális adatit egy-egy korábban tárolt, vagy akár más felhasználó adataival hasonlítják össze.

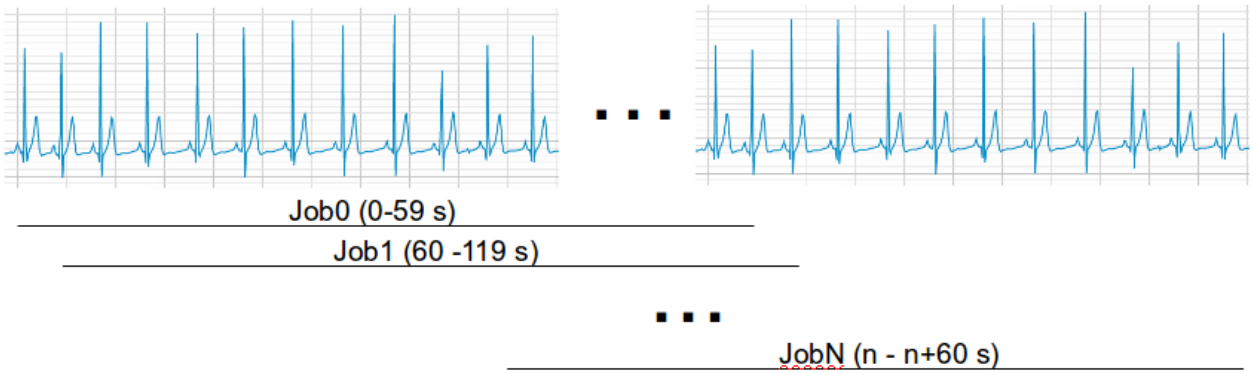
Az ECG adatok alapján történő pulzus számolásnak mindig futnia kell, ha érkezik ECG adat, és az éppen aktuális mérési eredményt kéne feldolgoznia.

Az ECG görbe tekintetében a pulzus az egy perc alatt mért értékekben megtalálható R csúcsok száma, a 2. ábrán egy-egy zöld körrel jelölve.



2. ábra: Pulzus adatok az ECG görbén

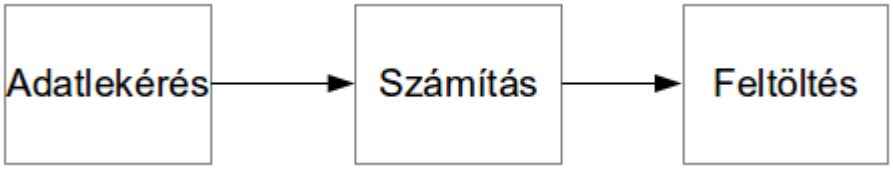
A feldolgozáshoz egy monitorozott felhasználótól folyamatosan érkező ECG adatokon egy perces ablakot definiálunk egy JOB (feladat) számára. Az ablakban található R csúcsok száma jelenti az ablak végéhez tartozó időpillanatban a pulzus értéket. A számítást egy másodperces csúszásokkal addig ismételjük, amíg van feldolgozandó adat, minden egyes másodpercben új JOB-ot indítva.



3. ábra: JOB-ok és ECG adatok viszonya

Pulzus számítás esetén az aktuálisan mért értékeken kell számításokat végezni és az eredményeiket rövid időn belül elérhetővé kell tenni a központi adatbázisban. Az algoritmus futási ideje nagyon rövid ezért egy-egy végrehajtás során kritikus a futtató middleware (számítási köztes réteg) okozta plusz terhelés időveszteségе. Sajnos ezen ok miatt nem tudtuk a többi algoritmus esetében alkalmazott grid-es végrehajtással megoldani a feladatot.

A pulzusmeghatározó alkalmazásunknak három feladatot kell elvégeznie.



4. ábra: Alkalmazás fő feladatai

A központi szerveren a 600Hz-es mintavételezésű ECG adatok ¼ perces egységekben vannak tárolva. Az adatlekérés során ezekből a csomagokból kapunk annyit, amennyiben benne van az az egy perc amire szükségünk van (4-6 csomag). HTTP protokollon kapunk egy egyperces 3 csatornás, 600Hz-es ECK adatrészletet, ennek valós mérete 2,5Mb +- 2% (5 rögzített csomag esetén). Az adatátvitel csökkentésének érdekében a kérést Accept-Encoding: gzip fejléc beállítással használjuk, így megtakarítjuk a hálózati sávszélesség 90%-át és még a be és kitömörítések után is nyerünk kérésenként egy pici időt (0,01 – 0,03 másodperc).

A számítási rész feladata hogy eldobja azokat a méréseket amelyek az általunk vizsgált egy perces időintervallumon kívül vannak, és a megmarasd adatok alapján meghatározza a pulzust.

A feltöltés szintén HTTP protokoll segítségével történik, itt nem alkalmazunk tömörítést mert az elküldendő adat mérete csak néhány byte.

Mivel ezen részfeladatok futásidejének az összege pár másodperc ezért a feladatokat az optimális végrehajtás érdekében egy JOB-ba foglaltuk.

Hasonló problémák a világban

Sok apró adat feldolgozásával a meteorológia és szeizmológiai területeken dolgozó kutatóknak napi rendszerességgel szembe kell nézniük. Szökőár [16] és a vulkáni aktivitás [19] előrejelzése, egyéb földmozgásokkal kapcsolatos szimulációk [17] [18] vizsgálata esetén az újabb publikációk tanulmányozásakor már gyakorlatilag csak valamilyen felhő alapú elsősorban saját egyedi megoldásokkal találkoztunk.

A kutatási ágazat mellett az ipar is egyre jobban a cloud rendszerek felé fordul. A Gartner előrejelzése [3] alapján 2012 után 213-ban is a cégek számára a harmadik legfontosabb technológia terület a cloud computing lesz.

1. Analytics and business intelligence
2. Mobile technologies
3. Cloud computing (SaaS, IaaS, PaaS)

5. ábra: Gartner 10 legfontosabb üzleti és technológiai terület 2013-ban [3]

A CIO Magazine idei technológiai felmérése alapján [4] a megkérdezett vállalati vezet?k 48%-a tervezi cége informatikai költségvetésének növelését, aminek várhatóan 36%-a cloud témakörben használják majd fel.

Célunk, hogy az általunk kidolgozott megoldás hamarosan egészségügyi szolgáltatásként legyen elérhető, ezért a nemzetközi kutatói és ipari trendek alapján a számításainkat a grid és szuperszámítógép alapokról a cloud felé irányítjuk. Ennek lesz első eleme az ECG alapú Pulzus meghatározás.

Felhősítés

Korábban más egészségügyi adatok feldolgozását már sikerült a WS-PGRADE/gUSE/DCI-Bridge rendszerrel megoldanunk, ezért a keretrendszer további felhasználása mellett döntöttünk, és azt is beköltöztettük a felhőbe. Ezzel nem volt semmi probléma, hiszen ez a rendszer könnyen telepíthető minden olyan környezetben, ami megfelel a telepítési leírásában lévő kritériumoknak.

A WS-Pgrade/gUSE/DCI-Bridge rendszer a Cloud-Broker platformon keresztül támogatja a clod rendszerek elérését, így ezen keresztül is lehetőségünk lett volna cloud erőforrásokat igénybe venni, de problémákba ütköztünk:

- A JOB-ok végrehajtási ideje elmarad az algoritmus természete és alkalmazási köre által elfogadható időktől. Egy JOB futtatása hasonlóan a grid-ekhez több percig is eltartott.

- A számítási inputok egy harmadik félen keresztül (Cloud-Broker svájci központja) jutnak el a tényleges végrehajtási helyre, ez egészségügyi adatok kezelése esetén ez akár jogi problémákhoz is vezethetne, amelyeket szerettünk volna elkerülni.
-

A Cloud rendszerek esetében a szolgáltatási szint (IaaS [5], PaaS [6], SaaS[7]) kiválasztásával a felhasználónak már az első pillanatban meg kell hoznia egy olyan döntést, ami akár kritikus is lehet a projektje számára. A problémát tovább fokozza, hogy már maga a SaaS, IaaS, PaaS definiciója sem egységes. A meghatározások túl sok értelmezési lehetőséget rejtenek magukban, amit a szolgáltatók ki is használnak, ezzel tovább nehezítve a felhasználók életét.

A megfelelő cloud szolgáltatási szint kiválasztásához először megvizsgáltuk, hogy jelenleg mit (grid, szuperszámítógép), és hogyan használunk fel az algoritmusok futtatása során.

Az eredmény meglepően egyértelmű lett: minden futtatási rendszer esetében egy olyan számítási platformot használunk, ahol az oda tervezett (fordított) tetszőleges alkalmazásainkat végre tudjuk hajtani. Ebből adódóan a feladatunkhoz egy PaaS megoldást kerestünk. Kerestünk, de nem találtunk! Megvizsgáltuk az elérhető nagyobb piaci szereplők PaaS megoldásait [8], [9], [10], [11], az ismertebb nyílt forráskódú változatokat [12],[13], és arra jutottunk, hogy a cloud világban a PaaS egy olyan környezetet jelent, amiben kifejleszthetünk a rendelkezésünkre bocsájtott platform által behatárolt webes megoldásunkat, felhasználva az előre telepített szoftvereket. Nekünk nem erre volt szükségünk, ezért lépnünk kellett a cloud piramison [14].

Első ránézésre a kényelmesebb választás a felfelé lépés a SaaS lett volna. Ebben az esetben:

- vagy minden alkalmazásunknak egy-egy külön virtuális gépet kellett volna készíteni, ami publikálja az alkalmazást mint egy szolgáltatás,
- vagy egy virtuális gépbe kellett volna berakni minden alkalmazást, amiket eltérő szolgáltatásokon keresztül tudnánk elérni.

Mivel az egészségügyi rendszerünkbe meg szerettük volna hagyni feldolgozási algoritmusok bővítésének lehetőségét, ezért a SaaS alkalmazása esetében fenn áll a

veszélye a virtuális gépek vagy az azon futó egymástól független szervizek élburjánzásának. Ezért inkább az IaaS-t vettük célpontba.

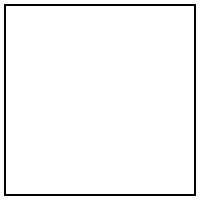
Egy infrastruktúra önmagában csak arra használható, hogy telepítsünk rá valamilyen szoftver rendszert, amivel valójában fogunk használni az erőforrásunkat. Nekünk egy ütemezőre, és a hozzá kapcsolódó számítási erőforrásokra volt szükségünk. IaaS környezetben tehát minket terhel az a munka, hogy kialakítsuk a számunkra valójában megfelelő PaaS-t vagy SaaS-t.

Mivel a végrehajtási idő számunkra kritikus ezért olyan ütemező megoldásra volt szükségünk, amin keresztül futtatva a legközelebb kerülünk a JOB-jaink valós végrehajtási idejéhez. Két lehetőséget vizsgáltunk meg:
1Telepítettünk egy PBS-t a saját cloud-unkba.
2Felhasználva a DCI-Bridge terheléselosztó képességét egy kis fürtbe szerveztünk néhány DCI-Bridge webalkalmazást.

Mérések

A méréshez az adatokat egy kosárlabda mérkőzésről vettünk. A pályán lévő egyik csapat játékosaira egy-egy ECG készüléket, és egy mobil telefont rögzítettünk. A telefonok a lokális WIFI hálózaton keresztül juttatták el az adatok a központi adatgyűjtő szerverre.

Mind a két mérési esetben az egyik csapat minden pályán lévő játékosnak dedikáltunk egy-egy processzort. A dedikált erőforrások ellenére sem számítottunk valós idejű feldolgozásra mert az algoritmusunkban lévő számítás ideje 0,8 és 1,2 másodperc körüliek voltak a szinte már elhanyagolható feltöltés és az 1,6 másodperces adatletöltési időt mellett.



5. ábra: Inputletöltések ideje egy szálon

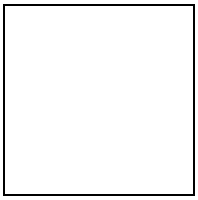
Mivel a másodpercenként induló egyperces pulzus adatok meghatározása több időt vesz igénybe mint az új adatok keletkezése, és a számítási erőforrások száma megegyezik az adatforrások számával nem lesz olyan időzítési problémánk hogy futásidőben egy JOB olyan adatot kér el amelyek még nem jöttek létre az ECG adatsora.

Egy kosárlabda csapat 5 játékosának egy 12 perces játékrészben 5x12x60=3600 pulzus meghatározását kell elvégezni. Ez ugyan ennyi JOB indítását jelenti, melyek összesen 2.160.000 ECG mérést dolgoznak fel. Ugyan ilyen körülmények között egy labdarúgó mérkőzésen 237.600 pulzus értéket kéne meghatározni, ha csak a pályán lévő játékosokat monitorozzuk.

Összehasonlítás

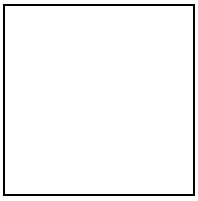
Mérések hiányában nem tudtuk felelős mérnöki döntést hozni, hogy a cloud környezetben melyik lenne számunkra a kedvezőbb megoldás, ezért a pulzus számolást egy végrehajtási idő mérésével egészítettük ki. Ennek segítségével az alkalmazás meg tudta mondani saját magáról, hogy valójában mennyi ideig futott.

PBS esetében egy pulzus kiszámítása átlagosan 3,981 másodpercet vett igénybe.



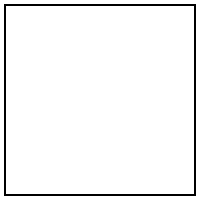
6.ábra: Futási idők alakulása PBS esetén

DCI-Bridge fürt esetén 3,58 másodperc volt az átlagos végrehajtási ideje az egy perces pulzus meghatározásának.



7. ábra: Futási idők alakulása DCI-Bridge esetén

Sajnos a gUSE nem támogatja az egy munkafolyamaton belüli JOB-ok idő alapú időzítését. Ahhoz, hogy a pulzus értéket ki tudjunk számolni szükséges az érintett időintervallum előre definiálása. Ez plusz JOB(ok) indítását igényli. A teljes folyamat ismeretében az elért futási eredmények alapján érdekes számításokat végezhetünk.

8. ábra: Folyamat végrehajtási idő 

A futásidő oszlop azt az időt tartalmazza, amit a gUSE rendszer szolgáltatott számunkra arról, hogy a workflow elindítása és befejezése között mennyi idő telt el.

A veszteség oszlopban az az érték látható, ami a tényleges workflow végrehajtása és a párhuzamos pulzus számításokra fordított idő különbsége. Ez az idő a gUSE használata érdekében létrehozott JOB(ok) futási ideje, és a futtató rendszer kommunikációs overhead-je.

Az eltérés oszlopban a központi adatgyűjtő szerver adatai alapján az első és utolsó pulzus adat megérkezése között eltelt idő található.

Összegzés

A felhő számunkra nem a megoldás szolgáltatta a számítási problémáinkra, csak egy eszközt adott, ahol saját magunk próbálhatunk ki, a saját magunk által összeállított végrehajtási alternatívák sorát. A felhő nem forradalmasította a JOB-jaink futtatását, hiszen még ebben az új környezetben is a már jól bevált eszközeinkre támaszkodva tudunk egyről a kettőre majd onnan remélhetőleg a háromra előre jutni.

A kitűzött célt elértük! Tudjuk a JOB-jainkat futtatni a felhőben. Esetünkben a pulzus adatok késése még mindig több mint amit szeretnénk, de ennek optimalizálása már egy másik konferencia témája lesz hamarosan.

Felhasznált hivatkozások listája:

[1] http://biotechweb.nik.uni-obuda.hu/web/

[2] http://guse.hu

[3] http://www.gartner.com/newsroom/id/2304615

[4]http://www.cio.com/article/721380/Mobile_BI_and_Cloud_Drive_Increased_IT_Spending

[5] http://www.gartner.com/it-glossary/infrastructure-as-a-service-iaas/

[6] http://www.gartner.com/it-glossary/platform-as-a-service-paas/

[7] http://www.gartner.com/it-glossary/software-as-a-service-saas/

[8] https://developers.google.com/appengine/

[9] http://www.windowsazure.com/en-us/

[10] http://aws.amazon.com/elasticbeanstalk/

[11] http://www.ibm.com/cloud-computing/us/en/paas.html

[12] http://www.heroku.com/

[13] https://openshift.redhat.com/app/

[14] http://www.saasblogs.com/images/uploads/2008/12/cloud_stack.gif

[15] http://www.ogf.org/OGF30/materials/2174/3+grid10-provisioning-ostermann.pdf

[16] V. N. Chebrov, A. A. Gusev, V. K. Gusyakov, V. N. Mishatkin, A. A. Poplavskii: Concept for developing a seismologic observation system for tsunami warning in the Russian Far East,
Seismic Instruments July 2010, Volume 46, Issue 3, pp 275-285

[17] Bettina P. Goertz-Allmann1 and Stefan Wiemer2: Geomechanical modeling of induced seismicity source parameters and implications for seismic hazard assessment, doi: 10.1190/?geo2012-0102.1 Geophysics January 2013 v. 78 no. 1 p. KS25-KS39

[18] M.Gabrin, E.Piolo: Seismic Event Recognition in the Trentino Area (Italy): Performance Analysis of a New Semiautomatic System, doi: 10.1785/?0220120025
Seismological Research Letters January/February 2013 v. 84 no. 1 p. 65-74

[19] Muhammad Farhan: Investigation of Volcanic Activity Monitoring Techniques for Aviation Safety, Publisher: Farhan Works 09/2011;