CompP2P: Sistema de Computació Distribuïda P2P

Iñigo Goiri Presa
Josep Rius Torrentó

Enginyeria Tècnica en Informàtica de Sistemes

Projecte dirigit per: Fernando Cores Prado i Francesc Solsona Tehàs

Departament d'Informàtica i Enginyeria Industrial

Escola Universitaria Politècnica
C/ de Jaume II, 69, *Campus Cappont*
E-25001 Lleida. Espanya

Telèfon: +34 973 70 27 03

Resum

Aquest projecte introdueix un nou sistema de computació distribuïda utilitzant una plataforma peer-to-peer (P2P). En aquest resum es presenta el disseny i la implementació d'aquest sistema de computació P2P anomenat CompP2P i basat en una topologia de peers purament descentralitzada.

A diferència d'altres sistemes amb una topologia centralitzada o amb una topologia hibrida, CompP2p no requereix cap tipus de servidor en la seva execució, sino que tots els nodes del sistema (anomenats peers d'aquí en endavant) s'organitzen ells mateixos automàticament i deleguen tasques a altres peers del sistema.

CompP2P intenta arribar al màxim nombre d'usuaris, fet que li permet prendre avantatge sobre altres solucions ja implementades.

1. Introducció

Durant les últimes dècades s'ha evidenciat una creixent demanda de recursos de còmput per part de multitud d'aplicacions intensives en còmput emmarcades en diferents àmbits d'investigació (ciències de la vida, models financers, química quàntica, astronomia i investigació espacial...). Aquestes aplicacions requereixen de supercomputadors composts per milers de processadors capaços de proporcionar una gran potència de còmput i una gran capacitat de transferència i gestió d'informació. Els sistemes informàtics capaços de proporcionar aquestes prestacions requereixen d'infraestructures de computació molt complexes, que tenen un elevat cost fet que impossibilita la generalització de la seva utilització en empreses i grups de recerca.

Per altra banda, el parc d'ordinadors personals existents a Espanya no ha parat de créixer durant els últims anys, gràcies a l'expansió de la informàtica tant en l'àmbit empresarial com en el domestic. Conjuntament a l'increment del nombre d'ordinadors, la seva potència de còmput també s'ha multiplicat, oferint prestacions cada vegada més elevades, que superen en la majoria dels casos els requisits de les aplicacions que utilitzem quotidianament. Aquest augment de potència ha vingut acompanyat per una considerable disminució del preu de producció del maquinari, fet que provoca en la majoria dels casos que la capacitat de còmput dels ordinadors personals estigui sobredimensionada respecte a les necessitats reals dels usuaris i les aplicacions que aquests utilitzen. Es genera, per tant, un desaprofitament de recursos, ja que romanen els sistemes informàtics la major part del temps ociosos o bé infrautilitzats.

D'aquesta forma, es dóna la paradoxa que mentre les empreses i els grups d'investigació cada vegada necessiten més recursos de còmput, el volum de recursos informàtics ociosos dels ordinadors personals no para de créixer. Per tant, nosaltres proposem utilitzar els recursos ociosos associats amb els ordinadors personals i reaprofitar-los per a executar aplicacions intensives en còmput.

En aquest treball s'ha fet una estimació de la potència de còmput prenent com referència el parc d'ordinadors personals existent és Espanya en el 2004 i assumint que s'assoleix integrar el 1% dels ordinadors. S'ha suposat que els ordinadors estaven equipats amb un AMD Athlon a 1530 Mhz, el qual proporciona una prestacions de còmput d'aproximadament 3 GFlops. Finalment, suposem que en terme mitjà els equips informàtics romanen el 50% del temps ociosos, i és justament aquest temps el qual nosaltres pretenem aprofitar. Si fóssim capaços d'explotar solament una petita part de les prestacions no utilitzades pels ordinadors personals assoliríem un sistema amb una potència de còmput de més de 200.000 GFlops. Aquestes prestacions únicament estaria al nivell de la màquina Blue Gene/L, la més potent del món, però amb una important diferència; mentre que Blue Gene/L té un cost d'uns 200 milions (sense comptar els costos de llum, manteniment i personal), la nostra alternativa, en canvi, sortiria pràcticament a cost zero (el propietari de la màquina es fa

càrrec de totes les despeses: cost d'adquisició, manteniment i llum).

L'objectiu d'aquest projecte és el disseny d'una plataforma de computació distribuïda que sigui capaç d'aprofitar aquests recursos ociosos dels ordinadors personals, proporcionant un sistema de computació d'altes prestacions amb un baix cost.

2. Motivacions del treball, punt de partida

Actualment hi ha diversos entorns de computació distribuïda que utilitzen Internet, com pot ser el Grid. Aquest tipus d'entorn però, presenta certs inconvenients, com la dificultat d'instal·lació i una jerarquia massa estricta i controlada.

És per això que neix la necessitat de desenvolupar una aplicació de còmput distribuït amb comunicació a nivell mundial, de fàcil instal·lació i sense jerarquies a la xarxa. Aquesta filosofia de treball és la que presenten les aplicacions que segueixen una arquitectura d'igual a igual (*peer to peer* o *P2P* en anglès). Aquestes xarxes defineixen un sistema de comunicació que no té clients ni servidors fixes, sinó una sèrie de nodes que es comporten alhora com clients i com servidors dels altres nodes de la xarxa, en el qual les dades es transfereixen a través d'una xarxa dinàmica. Aquest model contrasta amb el model client-servidor.

Actualment, les aplicacions basades en arquitectures entre iguals s'usen en la seva gran majoria per la compartició de dades (programes tant coneguts com emule, edonkey, kazaa... en són exemples) i no tant en la de còmput. D'altra banda, les alternatives en aquest camp són escasses, i les més destacades (com ara el càlcul del escalfament global del planeta) són per tasques concretes, perdent per tant el sentit de computació entre iguals, ja que la funció dels usuaris es limita a cedir recursos.

Aquest treball pretén donar una alternativa diferent a la computació distribuïda que s'utilitza actualment, creant un sistema fàcil d'usar.

Es vol aconseguir doncs, un sistema totalment entre iguals, i usable per tothom. Un sistema on tots els usuaris puguin dur a terme les mateixes tasques, no limitant-se només a aportar recursos sinó que també puguin delegar tasques a altres peers.

3. CompP2P

En aquesta secció s'explica de forma molt resumida el que fa referència amb l'arquitectura i el disseny de CompP2P.

Primerament, l'arquitectura del sistema treballa sobre la màquina virtual de Java [2]. Amb això s'intenta aconseguir una aplicació totalment multiplataforma capaç de funcionar en qualsevol ordinador amb independència de la seva arquitectura (i386, ppc, x86_64...) i del seu sistema operatiu (Windows, Unix, MacOS...).

En l'arquitectura podem distingir dos parts clares comunicades entre elles mitjançant sockets TCP. Aquestes dues parts estan representades en la figura 1.

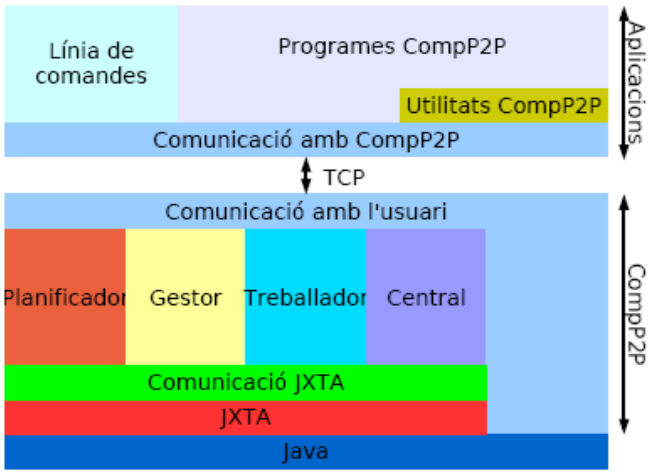

Figura. 1. Esquema de l'arquitectura de CompP2P

D'una banda, CompP2P té una part central que s'encarrega de gestionar la part del peer de treballador, de gestor i la planificació de tots els events. Aquesta capa s'encarrega també de donar comunicació amb l'usuari mitjançant sockets TCP.

En un segon terme, també s'han dissenyat mecanismes de comunicació amb el sistema per part de l'usuari, donant una interfície de comunicació simple però a l'hora ampliable, per tal d'un fàcil ús de CompP2P. Per donar a l'usuari uns mecanismes concrets d'utilització, també s'han desenvolupat utilitats d'alt nivell.

Amb l'objectiu d'aconseguir un disseny estructurat, s'ha elegit un disseny de l'aplicació en mòduls independents (representats a la figura 2). Amb això es pot implementar un mòdul per complet, sense necessitat de tenir implementat cap altre mòdul. La metodologia que s'ha seguit en totes les fases de desenvolupament de l'aplicació ha estat totalment orientada a objectes. S'ha intentat aconseguir un sistema poc acoplat i fortament cohesionat.

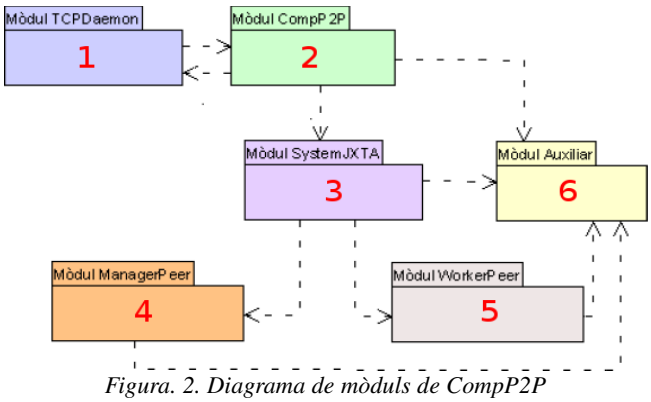

Figura. 2. Diagrama de mòduls de CompP2P

1. TCPDaemon: És el mòdul de comunicació amb l'usuari. S'encarrega de rebre els objectes de les aplicacions, serialitzar-los, enviar-los a un peer remot, esperar rebre'ls executats remotament, deserialitzar-los i finalment retornar-los a l'aplicació que havia fet la petició.

2. CompP2P: És el mòdul central de l'aplicació i s'encarrega de posar en marxa tot el sistema.
3. SystemJXTA: Aquest mòdul es pot definir com el gestor JXTA de CompP2P. La resta de l'aplicació intenta no tenir contacte amb aquesta plataforma de desenvolupament P2P per tal d'alliberar al programador de la càrrega que suposa haver de conèixer els detalls de JXTA.
4. ManagerPeer: La seva tasca principal es operar com a gestor d'un grup de peers.
5. WorkerPeer: Té la responsabilitat de gestionar els treballs que es volen dur a terme des d'aquest node i distribuir-los pel sistema.
6. Auxiliar: Són principalment estructures de dades que donen suport i faciliten la implementació de la resta del sistema.

Finalment, pel que fa a les eines de treball, per fer-lo s'ha utilitzat un editor de text pla i el compilador de java de sun en la versió 5.0. D'altra banda, el sistema està implementat sobre la plataforma JXTA [4] que ens proporciona serveis per tal de poder formar grups de peers, establir comunicacions entre ells, enviar-se missatges a traves de tallafocs i NATs... Quant a la documentació s'ha utilitzat l'eina nativa de java, javadoc, per dur a terme la API del programa. I pel que fa a la redacció de la memòria s'ha utilitzat LATEX.

4. Resultats obtinguts

Abans de presentar els resultats obtinguts, es farà una petita introducció per tal de descriure l'entorn de treball utilitzat en les proves, així com l'aplicació avaluada.

Característiques del entorn de treball

Per tal de posar a prova la multiplataformitat de l'aplicació, s'han desenvolupat les proves en un entorn de treball heterogeni. S'han utilitzat 30 Pentium 4 amb 512 MB de memòria RAM cada un, 12 dels quals amb una velocitat del processador de 2'8 GHZ i la resta de 2'4 GHZ.

Amb això s'aconsegueix tenir 30 peers amb diferents sistemes operatius, 2 Debian GNU/Linux i 28 Microsoft Windows 2000. Tots ells amb Java Runtime Environment Version 5.0 prèviament instal·lat.

Tasca a executar: Tanca òptima

L'aplicació calcularà quina és la combinació òptima d'arbres que s'han de tallar per fer una tanca al voltant dels arbres restants. Per definir un arbre, s'utilitza la seva ubicació, la longitud de tanca que es pot fer amb l'arbre i el seu valor econòmic.

La resolució d'aquest problema té un cost exponencial, cada arbre que afegim al càlcul implica duplicar el cost ja que ha de calcular les combinacions que ja hi havia més les resultants amb l'arbre afegit. És un problema fàcilment distribuïble. A cada peer del sistema se li assignarà unes combinacions per calcular i retorna d'aquestes la combinació òptima. Per últim cal avaluar tots el resultats parcials i retornar l'òptim.

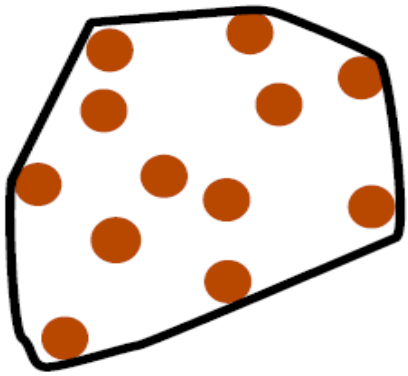

Figura. 3. Exemple tanca òptima

A la figura 3 es mostra un exemple del càlcul de la tanca òptima amb 12 arbres. Com a principal tret, es pot veure que tots els arbres que queden estan dins de la tanca.

Avaluació dels resultats (26 arbres i 30 particions)

A l'hora d'obtenir els resultats cal tenir en compte la plataforma d'execució. Ja hem explicat els avantatges de Java però al introduir una màquina virtual s'incrementa el temps d'execució. Aquesta sobrecarrega ha de ser mesurada per obtenir resultats reals.

Primerament compararem una versió sèrie de la nostra aplicació de proves, la tanca òptima, en Java i C++.

Això suposa una pèrdua d'un 45%, per tant podem confirmar i quantificar la baixada d'eficiència introduïda per la màquina virtual i la interpretació del codi.

Programa interpretat (Java): 15'43.345"
Programa compilat (C++): 10'59.911"

Un cop s'ha pres consciència de les limitacions de l'execució de la plataforma, cal mesurar el retard introduït pel nostre sistema de computació distribuïda, CompP2P. Anem a comparar la execució en sèrie sobre la màquina virtual de Java amb l'execució i l'execució en un sol peer amb l'arquitectura CompP2P (introduint la comunicació master-gestor-treballador).

Java execució en sèrie: 15'43.345"
CompP2P Overhead (30 particions): 18'11.106"

Escalablitat

Partint de la base de que Java és més ineficient que altres llenguatges de programació, es pot veure que amb la utilització de CompP2P amb dos màquines ja s'ha reduït el temps d'execució en 35 segons, el que significa una reducció d'un 5%. La millora no és gaire important però amb un número superior de peers aquesta diferència respecte a una versió paral·lela en C seria mínima.

Un cop vista la diferència bàsica respecte una versió amb C, s'analitza l'evolució dels temps d'execució segons el número de peers. A la gràfica de la figura 4 es pot observar que l'evolució dels temps és bastant propera a una situació ideal, aquesta situació es donaria en cas de que la feina fos

dividida entre tots els peers igualment, partint com a base el temps d'execució amb un únic treballador.

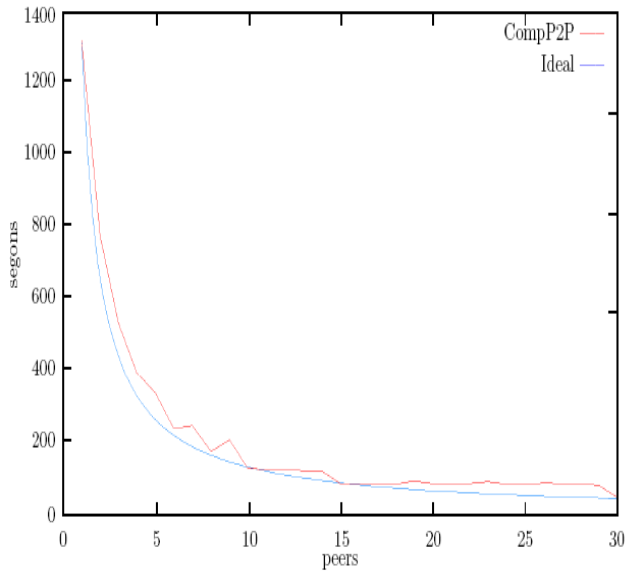

Figura 4. Gràfica dels resultats obtinguts

És important comentar que la distribució en 30 particions fixes fa disminuir l'eficiència del sistema per un cert número de peers. Tenint en compte que CompP2P disposa de la possibilitat de conèixer el número de treballadors que hi ha disponibles, es podria aprofitar aquesta possibilitat i dividir el treball en un nombre variable de tasques segons el número de peers treballadors disponibles. Amb això s'eliminarien els esglaonaments dels temps i aquests serien més propers als ideals en tot moment.

5. Conclusions

CompP2P és totalment independent del sistema operatiu que s'utilitzi. Aquesta característica és deu a la plataforma d'execució, Java. D'altra banda, la instal·lació del programa és molt senzilla i no necessita cap modificació en el nostre sistema, només es precisa la instal·lació de la màquina virtual de Java de Sun.

El fet de fer un sistema completament P2P comporta avantatges com pot ser la independència respecte a la resta de l'entorn de peers. El desenvolupament de l'arquitectura d'iguals amb JXTA ha estat certament complexa ja que aquesta plataforma dóna una visió a molt baix nivell de les comunicacions, creació de grups, manteniment, etc... També s'ha de dir que sense aquesta plataforma la implementació de CompP2P hagués estat pràcticament impossible degut a la seva complexitat.

Per concloure podríem dir que inicialment no s'havien plantejat objectius pel que fa al rendiment del sistema CompP2P, però un cop realitzades les proves s'ha pogut veure que tenint en compte les limitacions que té Java respecte a llenguatges compilats, el rendiment ha estat força satisfactori, i com s'ha demostrat a l'apartat d'experimentació, és un sistema amb una escalabilitat molt elevada.

6. Agraïments

Volem agrair a Jordi Viladrich Pàmpols pel seu recolzament i col·laboració rebuda de la seva part. I, molt especialment, mostrem el nostre agraïment als Directors del treball final de carrera, Dr. Fernando Cores Prado i Dr. Francesc Solsona Tehàs, per les seves oportunes aportacions.

Referències

[1] Java 1.4.2 API: *http://java.sun.com/j2se/1.4.2/docs/api/*

[2] Instalació de Java 1.5:

http://www.java.com/en/download/help/5000010500.xml

http://www.java.com/en/download/linux_manual.jsp

http://www.java.com/en/download/help/5000010300.xml

[3] JXTA 2.3.7 API:

http://platform.jxta.org/nonav/java/api/index.html

[4] JXTA:

http://www.jxta.org/docs/JxtaProgGuide_v2.3.pdf

http://www.brendonwilson.com/projects/jxta-book/

[5] JNGI: *http://jngi.jxta.org/*

[6] N. Griffiths, K.-M. Chao, and M. Younas, "Fuzzy Trust for Peer-to-Peer systems", 26th International Conference on Distributed Computing Systems (ICDCS 2006), 2006.

[7] Wei Wu, Phu Dung Le: "An Efficient and Secure Code Sharing for Peer-to-Peer Communications", International Conference on Information Technology: New Generations, pp. 476-481, 2006.

[8] A. Mondal, M. Kitsuregawa, "Privacy, Security and Trust in P2P environments: A Perspective", 3rd Int. Workshop on P2P Data Management, Security and Trust, pp. 682-686, 2006.

[9] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-peer computing", Technical Report HPL-2002-57, HP Lab, 2002.

[10] N. Drost, R.V. van Nieuwpoort, H. Bal, "Simple Locality-Aware Co-allocation in Peer-to-Peer Supercomputing", Proc. of the 6th IEEE Int. Symposium on Cluster Computing and Grid Workshops (CCGRIDW?06), 2006.

[11] A. Auvinen, M. Vapa, M. Weber, N. Kotilainen and J. Vuori, "Chedar: Peer-to-Peer Middleware", Proc. of the 20th IEEE Int. Parallel and Distributed Processing Symposium (IPDPS'06), 2006.

[12] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing", Proc of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), 2003.

[13] M. Venkateswara Reddy, A. Vijay Srinivas, Tarun Gopinath, D. Janakiram, "Vishwa: A reconfigurable P2P middleware for Grid Computations", Proc. of the International Conference on Parallel Processing (ICPP'06) pp. 381-390, 2006.

[14] D. Talia, P. Trunfio, "Towards a synergy between P2P and Grids", IEEE Internet Comput. 7 (4), pp. 94-96, 2003.