MONITORITZACIÓ DE L'EXECUCIÓ DE TASQUES A LACOLLA

- MEMÒRIA -

Universitat Oberta de Catalunya

Treball Fi de Carrera dels estudis de l'Enginyeria Tècnica en Informàtica de Gestió

Estudiant: Juan Francisco Lahuerta Martinez
Consultor: Antoni Ribes Sànchez
Lliurament: 14 Gener 2005

RESUM

LaCOLLA [Marquès 2003] és una infraestructura descentralitzada i distribuïda que ofereix, a les aplicacions, funcionalitats de col·laboració. Dins d'aquest conjunt de funcionalitats hi ha la d'execució de tasques, que permet que els iguals puguin executar tasques usant els recursos computacionals dels grups on pertanyen.

LaCOLLA, però, no incorpora cap eina o mecanisme que ajudi a controlar i observar l'estat tant de les tasques com dels iguals que les executen, per tal de garantir que es dur a terme d'una manera eficient, però també que no es veu afectat el rendiment de les aplicacions dels iguals.

En aquest treball s'estudien els mecanismes necessaris, que ofereixin a les aplicacions i al sistema, informació relacionada amb l'execució de les tasques, dels recursos que està consumint, càrregues dels iguals, etc. Una de les utilitats que té, consisteix en la possibilitat de detectar tasques que estan tenint problemes o que estan saturant un membre del grup, i actuar en conseqüència, eliminant o aturant durant un temps una tasca ó delegar la seva execució a un altre membre del grup amb més disponibilitat, etc.

Es proposa una arquitectura que ofereix dos tipus de serveis ben diferenciats:

- Serveis de consulta. Aquests serveis ofereixen informació relativa a les tasques i els iguals. Per exemple, permeten construir una aplicació per poder visualitzar l'estat de les tasques i els iguals d'un determinat grup o de tots els grups.

- Serveis de recol·lecció de dades de monitorització, detecció i recuperació de fallides. Són serveis autònoms del sistema que garanteixen l'execució de les tasques, per la qual cosa no hi ha cap interacció directa de l'exterior del sistema. La seva funció principal és la de recol·lectar la informació necessària de les tasques i la carrega dels iguals, per tal de detectar possibles fallides i poder recuperar-se. Aquests mecanismes de detecció i recuperació permeten als membres del grup, abstraure-los del seguiment de la tasca.

Per a validar l'arquitectura s'ha construït un prototipus en el que s'han implementat els mecanismes necessaris per a poder simular diverses situacions. Es deixa per a un treball futur completar aquest prototipus amb la resta de mecanismes definits, validant la seva correcció o proposant-ne d'altres.

ÍNDEX

CAPÍTOL 1 - INTRODUCCIÓ

EL PROBLEMA

LaCOLLA [Marquès 2003] és una infraestructura descentralitzada i distribuïda que ofereix, a les aplicacions, funcionalitats de col·laboració. Dins d'aquest conjunt de funcionalitats hi ha la d'execució de tasques, que permet que els iguals puguin executar tasques usant els recursos computacionals dels grups on pertanyen.

LaCOLLA, però, no incorpora cap eina o mecanisme que ajudi a controlar i observar l'estat tant de les tasques com dels iguals que les executen, per tal de garantir que es dur a terme d'una manera eficient, però també que no es veu afectat el rendiment de les aplicacions dels iguals.

Sembla clara la necessitat de disposar d'uns mecanismes a LaCOLLA, que ofereixin, a les aplicacions i al sistema, informació relacionada amb l'execució de les tasques, dels recursos que està consumint, càrregues dels iguals, etc. Una de les utilitats que té, consisteix en la possibilitat de detectar tasques que estan tenint problemes o que estan saturant un membre del grup, i actuar en conseqüència, eliminant o aturant durant un temps una tasca ó delegar la seva execució a un altre membre del grup amb més disponibilitat, etc.

OBJECTIU DEL TREBALL

Els objectius que es volen aconseguir en aquest treball són:

- Conèixer LaCOLLA i la problemàtica dels sistemes peer-to-peer, distribuïts i descentralitzats en l'execució de tasques.

- Identificar i analitzar quins mecanismes són necessaris per a dur a terme la monitorització de l'execució de les tasques a LaCOLLA.

- Identificar quins mecanismes són necessaris a LaCOLLA, per a la bona gestió dels recursos del grup: Com la migració de tasques entre iguals, cancel·lació de tasques, etc...

- Definició de la interfície que han d'implementar les tasques per a poder interactuar amb elles, amb els requeriments que es demanen en aquesta infraestructura (grups heterogenis, independència de la plataforma, etc...)

ABAST DEL TREBALL

El resultat final d'aquest TFC serà la implementació del sistema de monitorització de tasques a la versió actual de LaCOLLA. Aquest sistema ha de permetre a les aplicacions que utilitzen LaCOLLA, mitjançant la interfície que s'haurà de definir, obtenir informació relativa a les tasques en execució i dels iguals que les executen. El sistema oferirà, a grans trets, les següents característiques:

- Conèixer l'estat de les tasques que s'estan executant a LaCOLLA i obtenir informació de cada tasca en execució (Temps d'execució, CPU usada, memòria utilitzada, etc...).

- Canviar la prioritat, cancel·lar, pausar o reanudar qualsevol tasca que s'estigui executant en un moment donat.

- Conèixer l'estat dels iguals i la carrega d'aquests. El sistema executarà 'sensors' a les maquines dels iguals, on es monitoritzarà la CPU, memòria i xarxa. Aquests sensors permetran al sistema de monitorització detectar tasques que estan tenint problemes o que estan saturant un igual. En aquests casos el sistema,de manera autònoma, reubicarà l'execució en un altre igual ó aturarà o cancel·larà la tasca durant un temps determinat.

- El sistema de monitorització no incidirà en el funcionament normal de LaCOLLA.

A part de la utilitat como eina de monitorització, aquest sistema pretén proveir amb informació que permeti, per exemple, al TDA (Task Dispatcher Agent) seleccionar l'Executor Agent més adient per la tasca a executar.

ENFOCAMENT I METODOLOGIA

En iniciar aquest TFC es tenia clar que es volia que el sistema de execució de tasques de LaCOLLA tingués uns mecanismes que permeteren detectar problemes a l'execució de les tasques i recuperarse d'aquests, però no s'havia definit quins havien de ser aquests mecanismes.

En aquest sentit, la primera part d'aquest TFC ha consistit en fer una investigació per comprovar l'estat de l'art i decidir quines funcionalitats eren interessants d'incorporar al sistema de monitorització de tasques de LaCOLLA. Un cop definit l'àmbit d'aquest treball, la segona part ha consistit en desenvolupar un prototipus per a validar l'arquitectura proposada.

Aquest treball ha seguit els models proposats a l'enginyeria del programari i tècniques de desenvolupament de programari, estudiades a les assignatures de Enginyeria del programari i Tècniques de Desenvolupament del Programari de la UOC.

PLANIFICACIÓ DEL TREBALL

Per tal de garantir l'assoliment final dels requeriments d'aquest treball, s'han definit un seguit de fites que han marcat el ritme de treball.

Fita 1. Pla de Treball
Aquest fita té com a límit el dia 22 de setembre. Servirà de cloenda de la fase preliminar i el document més important serà el Pla de Treball que s'haurà de trametre en aquesta data.

PAC 1.
Requeriment de l'assignatura. L'entregable és el pla de treball.

Fita 2. Proposta definitiva
La data és 12 d'octubre. En aquest punt ja s'hauran pres totes les decisions que afectaran a les característiques de la monitorització. Es determinarà l'abast del treball.

Fita 3. Especificació
La data és el 28 d'octubre. En aquest punt quedaran definides les funcionalitats del subsistema de monitorització de l'execució de tasques.

PAC 2.
Requeriment de l'assignatura. S'ha establert la data del 2 de novembre i es lliurarà un document amb l'especificació.

Fita 4. Disseny
Aquesta fita marca la fi de l'etapa de disseny. La data és el 25 de novembre.

PAC 3.
Requeriment de l'assignatura. S'ha establert la data del 9 de desembre i es lliurarà un document amb el disseny i part de la implementació que s'estarà fent.

Fita 5. Implementació i Validació
La implementació haurà d'estar finalitzada i validada pel 29 de desembre.

Fita 6. Lliurament final
Aquesta serà la darrera fita del TFC i marcarà el final d'aquest treball. Abans del 14 de gener es farà el lliurament final compost per la Memòria amb la presentació virtual.

PRODUCTES OBTINGUTS

El prototipus desenvolupat per tal de fer la validació de l'arquitectura s'ha construït tenint en compte que ha de poder ser la base per al desenvolupament d'una ampliació de les funcionalitats. En aquest sentit, el llenguatge de programació emprat ha estat Java, ja que el prototipus de LaCOLLA que es prenia com a punt de partida utilitza aquest llenguatge. Com que la validació de l'arquitectura s'ha fet, per raons logístiques, en un entorn Windows, s'ha fet una implementació dels components nucli del sistema (els sensors de tasques i iguals), basats en aquesta plataforma. Com a entorn de treball s'ha utilitzat Eclipse, per les facilitats que ofereix per a treballar amb projectes Java de certa envergadura.

El producte obtingut, doncs, és un projecte Java que permet seguir amb el desenvolupament del sistema, dur a terme simulacions, generar versions, etc.
El projecte inclou les classes pròpies de LaCOLLA, fonts i binaris, i una implementació nativa per Win32 dels components nucli de l'arquitectura

ORGANITZACIÓ DE LA MEMÒRIA

El primer capítol de la memòria defineix els objectius, l'abast i el marc de treball d'aquest TFC. La resta de capítols presenten l'arquitectura proposada i la seva validació.

En el capítol 2 es defineixen els requeriments que ha de complir el sistema de monitorització de tasques a LaCOLLA.

En el capítol 3 es proposa i descriu l'arquitectura del sistema, definint els nous components i les modificacions sobre els existents.

En el capítol 4 es detalla com s'ha validat l'arquitectura proposada, quines funcionalitats s'han implementat en el prototipus i com s'han dut a terme les proves i els seus resultats.

Per últim, en el capítol 5 es presenten les conclusions i es comenten ampliacions futures d'aquest treball, que no s'han contemplat en el disseny actual però que pot ser interessant d'incloure-les en el futur.

CAPÍTOL 2 - RECOLLIDA DE REQUISITS

REQUISITS

El sistema de Monitorització de tasques ha de encarregar-se de monitoritzar les tasques que estan en execució a LaCOLLA, controlant i observant l'estat tant de les tasques com dels iguals que les executen, per tal de garantir que es dur a terme d'una manera eficient, però també que no es veu afectat el rendiment de les aplicacions dels iguals ni el funcionament normal de LaCOLLA. L'arquitectura de la nova funcionalitat cal que sigui conseqüent amb els requeriments actuals de LaCOLLA, i per tant ha de tenir en compte els següents aspectes:

- Tots els membres del grup han de saber en tot moment com està evolucionant el grup i han de tenir accés a la darrera versió de la informació generada en el grup.

- La infraestructura ha de ser autoorganitzada. Entenent per autoorganització l'habilitat d'un sistema per a reorganitzar els seus components sense requerir la intervenció de cap element extern.

- Els grups han de funcionar amb els recursos propis, això vol dir que siguin autosuficients que no depenguin de cap autoritat d'administració que els proporcioni els recursos per a poder dur a terme la seva activitat.

- La infraestructura ha de ser descentralitzada. Seguint el model de les arquitectures d'igual a igual es vol que no hi hagi cap node del grup que actuï de coordinador del grup. Es vol que cada node sigui autònom.

El sistema ha de permetre obtenir la següent informació de qualsevol tasca en execució (independentment del Sistema Operatiu que executa la tasca):

- Data d'inici de l'execució
- Temps que porta en execució
- Temps de CPU (acumulada) que ha consumit.
- Percentatge de CPU actual
- Memòria que està usant
- Prioritat
- Estat de la tasca a la CPU
- Espai de disc assignat a la tasca

Pel que fa als iguals, el sistema ha de proporcionar informació per determinar la carrega d'un determinat igual. Per aquest motiu, el sistema proporcionarà la següent informació:

- Informació de la màquina de l'igual:
  - Numero de processadors
  - Tipus de processadors
  - Memòria instal·lada
  - Sistema operatiu instal·lat

- Informació de carrega:
  - Percentatge de CPU, en ús, actual
  - Memòria usada
  - Percentatge de Memòria en ús
  - Numero de processos en execució
  - Espai de disc disponible

La informació proporcionada pel sistema de Monitorització, ha de permetre als TDA (Task Dispatcher Agent) seleccionar els EA (Executor Agent) amb més disponibilitat, a l'hora d'assignar l'execució d'una tasca.

El sistema de Monitorització ha de dependre d'una aplicació o usuari per detectar tasques que estan tenint problemes o que estan saturant un membre del grup, i actuar en conseqüència, eliminant o aturant-les durant un temps o migrar-les a un membre del grup amb més disponibilitat.

Cal que els iguals puguin definir quina quantitat dels seus recursos computacionals volen aportar al grup. El sistema ha de garantir que no es satura l'igual, utilitzant més recursos dels que l'igual ha aportat.

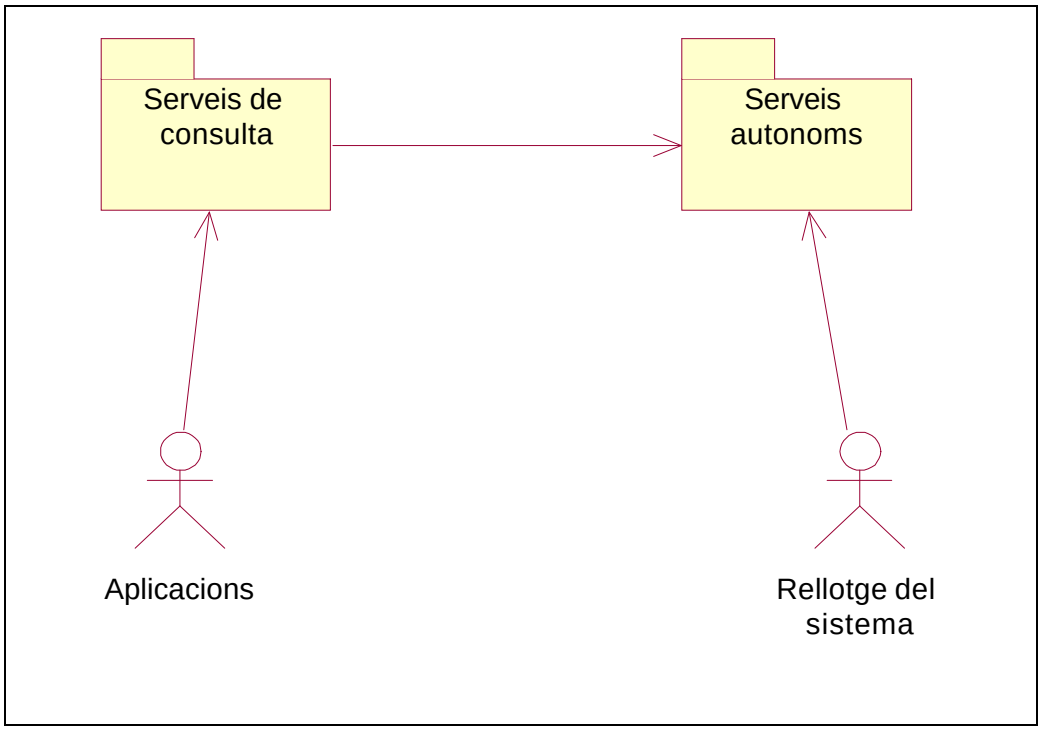
ESPECIFICACIÓ

El sistema de Monitorització de l'execució de tasques, a partir d'ara SMET, ofereix dos tipus de serveis ben diferenciats:

- Serveis de consulta. Els actors principals d'aquests serveis són les aplicacions. Aquests serveis ofereixen informació relativa a les tasques i els iguals. Per exemple, permeten construir una aplicació per poder visualitzar l'estat de les tasques i els iguals d'un determinat grup o de tots els grups.

- Serveis de recol·lecció de dades de monitorització, detecció i recuperació de fallides. Són serveis autònoms del SMET que garanteixen l'execució de les tasques, per la qual cosa no hi ha cap interacció directa de l'exterior del sistema. La seva funció principal és la de recol·lectar la informació necessària de les tasques i la carrega dels iguals, per tal de detectar possibles fallides i poder recuperar-se. Aquests mecanismes de detecció i recuperació permeten als membres del grup, abstraure-los del seguiment de la tasca.

El diagrama següent mostra la composició del sistema en paquets de serveis. Es pot apreciar com els serveis de consulta depenen dels serveis autònoms de SMET. Bàsicament de la funcionalitat de recol·lecció de dades.

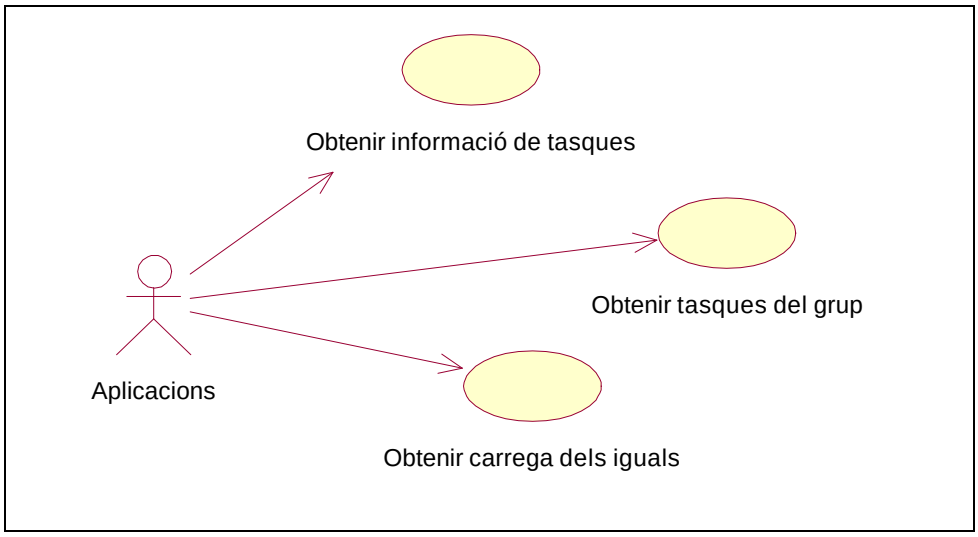


Il·lustració 1 - Diagrama de paquets

A continuació es detallen i s'especifiquen les funcionalitats que ofereixen els paquets de consulta i de serveis autònoms de SMET.

SERVEIS DE CONSULTES

Els serveis de consulta permeten a una aplicació consultar la informació completa d'una determinada tasca, així com informació relacionada amb la carrega d'un determinat igual. El següent diagrama mostra els casos d'us d'aquest paquet de serveis.



Il·lustració 2 - Diagrama de casos d'us del paquet de serveis de consulta

CAS D'US: OBTENIR LES TASQUES EN EXECUCIÓ D'UN GRUP

Descripció	Servei que proporciona les tasques en execució o pendents d'execució d'un determinat grup de LaCOLLA.
Actors implicats	Aplicació de LaCOLLA

Flux normal

L'aplicació proporciona l'identificador del grup del qual vol obtenir les tasques que s'estan executant o que estan pendents d'execució, mitjançant el API de LaCOLLA. El sistema retornarà a l'aplicació una llista amb totes les tasques registrades al grup. Si no hi ha cap tasca registrada, llavors la llista proporcionada estarà buida.

CAS D'US: OBTENIR INFORMACIÓ DE CARREGA D'UNA TASCA

Descripció	Servei que proporciona informació de monitorització de una determinada tasca.
Actors implicats	Aplicació de LaCOLLA

Flux normal

L'aplicació sol·licita al sistema la informació de una determinada tasca. L'aplicació especifica l'identificador de la tasca a consultar. El sistema retornarà la següent informació relativa a la tasca especificada:

- Data inici de l'execució
- Temps que porta en execució
- Temps de CPU (acumulada) que ha consumit.
- Percentatge de CPU actual
- Memòria que està usant
- Prioritat
- Estat de la tasca a la CPU (Si esta suspended, running, etc...)
- Espai de disc assignat a la tasca

CAS D'US: OBTENIR INFORMACIÓ DE CARREGA DELS IGUALS

Descripció	Servei que proporciona informació de carrega dels iguals d'un determinat grup.
Actors implicats	Aplicació de LaCOLLA

Flux normal

L'aplicació proporciona l'identificador del grup del qual vol obtenir la informació de carrega dels iguals que hi pertanyen. El sistema retornarà a l'aplicació, per cada igual del grup, la següent informació:

- Percentatge de CPU, en ús, actual
- Memòria usada
- Percentatge de Memòria en ús
- Numero de processos en execució
- Espai de disc disponible

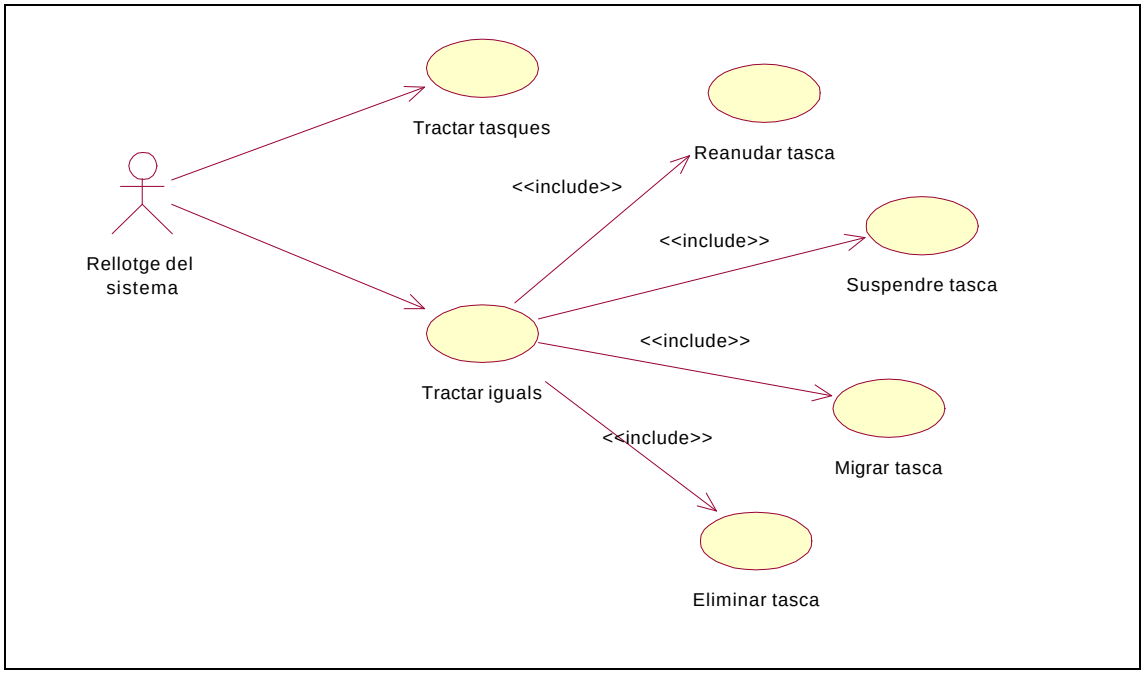
SERVEIS DE AUTÒNOMS

Els serveis autònoms són un conjunt de funcionalitats autònomes que proveeix el SMET. Aquests serveis garanteixen l'execució de les tasques, aplicant uns mecanismes per detectar fallides i recuperar-se d'aquestes.

Per tal de que aquests mecanismes facin una bona feina, cal alimentar-los amb les dades de carrega que es recullen dels iguals, així com de la informació de les tasques que executen els iguals. Aquestes dades son proporcionades pels sensors del SMET, que s'executen a cada igual. Aquest sensors monitoritzen els recursos que aporta l'igual (CPU, memòria, disc, etc...) generant un senyal, quant el recurs que monitoritzen sobrepassa un cert límit. Per exemple, un igual pot oferir només el 50% de la seva CPU, ja que la CPU la vol usar per algun altre tema que no sigui LaCOLLA. En aquest cas, el sensor generarà un senyal quan el nivell de la CPU d'aquest igual sobrepassi el 50%.

La informació generada pels sensors serveix per predir i detectar els problemes a l'execució de les tasques. Una vegada que el sistema s'adona de que hi ha o hi haurà un problema a un determinat igual (el sensor de CPU s'ha disparat, no hi ha espai suficient al disc, etc...) pausarà les tasques necessàries per fer 'apagar' el sensor. Si passat un temps determinat, no es pot reanudar l'execució de les tasques que s'han posat en pausa, iniciarà la migració d'aquestes a un altre igual amb més disponibilitat. Si aquesta migració no es pot donar en un temps determinat, s'eliminaran, notificant-lo a les aplicacions propietàries de les tasques.

El següent diagrama mostra els casos d'us d'aquest paquet de serveis:



Il·lustració 3- Diagrama de casos d'us del paquet de serveis autònoms

CAS D'US: TRACTAR IGUALS

Descripció	Detecció de problemes a l'igual
Actors implicats	Rellotge del sistema

Flux normal

Periòdicament és comprova si els sensors associats a l'igual han generat algun senyal. En aquest cas, s'inicia el procés de recuperació de les tasques implicades.

CAS D'US: TRACTAR TASQUES

Descripció	Recuperació de les tasques amb problemes a l'igual
Actors implicats	Rellotge del sistema

Flux normal

Es tracta la recuperació de les tasques, tal i com s'ha descrit anteriorment.

CAPÍTOL 3 – ARQUITECTURA DEL SISTEMA

INTRODUCCIÓ

La monitorització de les tasques es tracta d'un mecanisme intern de LaCOLLA, que a partir d'informació sobre les tasques que s'estan executant i de la càrrega dels iguals, es determina si una determinada tasca ha fallat o esta tenint problemes d'execució a l'EA assignat. Definim primer el concepte de tasca dins LaCOLLA i quina es la seva implementació actual.

Una tasca a LaCOLLA té una especificació d'execució, que defineix els requeriments per la seva execució: Recursos d'entrada (fitxers), plataforma d'execució, classe i mètode que implementa la tasca. Actualment, l'única plataforma suportada és Java. Per executar tasques en Java, els EA utilitzen els javaWorker, responsable de l'execució física de la tasca. Per executar-la, els javaWorker invoquen, des d'un thread al javaWorker, al mètode de la classe especificada en els requeriments de la tasca. La tasca finalitza quan el mètode invocat retorna el control o es produeix una excepció a la seva execució. En ambdós casos, s'informa del resultat de la tasca.

Des d'un punt de vista de monitorització de procés, totes les tasques que pot executar un EA, pertanyen a un únic procés: La Java Virtual Machine. Per aquest motiu, la informació referent a CPU i memòria associada a una tasca, per la plataforma Java, serà realment la del procés de la JVM que està executant l'EA. En aquest treball ens abstraurem de la plataforma d'execució, ja que podríem modificar el javaWorker actual i en lloc d'invocar la tasca en la JVM actual, que ho fes a una nova JVM (en un nou procés). Una altre raó és que segurament LaCOLLA sigui ampliada en futurs treballs, on s'incloguin altres plataformes d'execució.

Cal destacar, que si una tasca ha fallat, el javaWorker rep una excepció Java, indicant la fallida i el motiu (el stack trace). Aquestes dades s'adjunten al missatge de resultat de finalització de la tasca. Per aquest motiu, crec que el comportament actual davant aquest escenari ja és el correcte i el sistema de monitorització de tasques no ha de tractar aquest cas. La tasca continuaria fallant a qualsevol igual.

Tenint en compte el que s'ha comentat anteriorment, a continuació es descriu quina informació es necessita per determinar si una tasca tindrà o està tenint problemes amb la seva execució, així com la informació de càrrega que el sistema ha de publicar d'un igual.

Els factors que influeixen generalment en la correcta execució d'una tasca són: la CPU, la memòria i l'espai de disc de l'igual que l'executa. Si l'igual esta saturat, perquè hi ha massa processos executant-se, que provoquen que no hi hagi molta CPU i/o memòria disponible, provocarà que les tasques que l'igual executi, vagin lentes i inclús deixin inutilitzable l'igual. En aquesta situació es fa necessari disposar d'un mecanisme que defineixi quines tasques, dins d'un igual, es suspenen, es cancel·len o es migren a un altre igual amb major disponibilitat computacional, es a dir que tingui menys càrrega. Actualment no existeix un sistema de prioritats a les tasques que executen els EA, ni requeriments de recurs de còmput. Aquest treball contemplarà aquests nous requeriments, indicats en el document XML d'especificació per l'execució d'una tasca.

ARQUITECTURA PROPOSADA

El sistema de monitorització de l'execució de tasques, s'ha dissenyat en base a informació proporcionada per "sensors". Un sensor és un element del sistema, que observa les tasques i els recurs computacionals dels iguals,generant events de monitorització. Per exemple, aquest sistema proporciona sensors per monitoritzar l'ús de CPU, memòria i disc. Mitjançant aquests sensors, es controla la quantitat de recursos que cada igual ofereix al grup,i d'aquesta manera poder detectar sobrecàrregues als iguals, i proveir de la informació necessària (les mesures dels recursos), per poder recuperar els iguals sobrecarregats.

Amb aquest objectiu s'ha ampliat l'arquitectura actual de LaCOLLA, proporcionant noves responsabilitats als TDA i EA. També s'ha ampliat la funcionalitat dels UA, per suportar els serveis de consulta que s'ofereixen.

Les noves responsabilitats de cada component es detallen a continuació:

Noves responsabilitats dels **EA** (Executor **A**gent)

- Recol·lectar informació de càrrega de l'igual on s'està executant.
- Recol·lectar informació dels recursos de l'igual, que consumeix cada tasca que executa.
- Determinar si les tasques sobrepassen els recursos aportats per l'igual.
- Trencar la sobrecàrrega de l'igual, suspenent les tasques necessàries, garantint així, la seguretat del sistema de l'igual.

- Rebutjar les assignacions d'execució d'una tasca per sobrecàrrega de l'igual.

- Tenir en compte les prioritats de les tasques, a l'hora de realitzar les operacions sobre aquestes.

- Publicar la informació de càrrega de l'igual als TDA.

Noves responsabilitat dels **TDA** (Task Dispatcher Agent)

- Seleccionar els EA candidats a executar una determinada tasca, en base a la informació de càrrega proporcionada pels EA.

- Migrar les tasques a altre igual amb major disponibilitat de recursos computacionals.

- Mantenir la informació de carrega dels EA connectats al grup.

- Tenir en compte les prioritats de les tasques, a l'hora de realitzar les operacions sobre aquestes.

Noves responsabilitat dels **UA** (User Agent)

- Permetre a les aplicacions, tenir accés als serveis de consulta que proporciona el sistema de monitorització.

- Encaminar les noves peticions de consulta de les aplicacions cap a un TDA del grup.

Aquesta arquitectura modifica el javaWorker actual, amb la finalitat de abstraure la monitorització a aquest nivell, es generalitza l'execució de les tasques mitjançant un nou component: Worker. Aquest component esta modelat com una superclasse, on javaWorker i qualsevol futur Worker de plataforma, han de heretar. D'aquesta manera es permet abstraure l'implementador dels Workers del sistema de monitorització de les tasques, que aquests executen.

A continuació es descriu detalladament el sistema proposat mitjançant aquesta arquitectura.

DISSENY DE L'ARQUITECTURA

La CPU, memòria i espai de disc són els factors principals que influeixen en la correcta execució d'una tasca en un igual. Es fa necessari disposar d'uns mecanismes, ubicats als EA, que permeten determinar la càrrega de l'igual on s'executen, així com els recursos de l'igual que estan consumint les tasques que aquests executen.

La informació relativa a la càrrega de l'igual on s'executa l'EA, serà recol·lectada per un **PeerMonitor**. Aquest component està associat a cada EA. Quan l'EA sigui creat, aquest crearà un PeerMonitor que li proporcionarà la informació de carrega de l'igual. Quan finalitzi l'execució d'un EA, també finalitzarà la del PeerMonitor. Aquest component compleix amb les funcionalitats d'un **sensor**. Recol·lecta informació de CPU, memòria i espai de disc del directori de treball, enviant un senyal quan es sobrepassi un **llindar** configurat. Es pot donar el cas en que la CPU puji durant un petit espai de temps (1 o 2 segons), no apreciant-se constància. Per evitar l'enviament de senyals "no alarmants", el sensor només enviarà un senyal d'alerta quant el llindar està sobrepassat durant un temps mínim de 5 segons. Els llindars i el temps mínim per generar el senyal d'alarma en un sensor, és configurable mitjançant el fitxer de configuració.

Mitjançant els llindars dels sensors, l'usuari pot especificar la quantitat de recursos de l'igual (CPU, memòria i disc) que desitja oferir al grup. Per exemple, l'igual només ofereix el 50% de la seva CPU, 1GB de disc i 128MB de memòria.

En aquest treball, el PeerMonitor es compon de tres sensors: el de CPU, de memòria i disc. PeerMonitor utilitza el patró de disseny Composite, i permet afegir futurs sensors a la monitorització d'un igual, com podria ser: ample de banda de xarxa, fallides d'un disc, etc...

Els PeerMonitor utilitzen el patró Observable per notificar els senyals. Quan un sensor s'ha activat, s'ha sobrepassat el llindar configurat, ho notifica als Observer, l'EA associat, que estan subscrits al PeerMonitor. D'aquesta manera l'EA és conscient de que l'igual està sobrecarregat.

El sistema de monitorització també proporciona informació sobre els recurs que estan consumint les tasques que hi ha en execució. Aquesta informació es recol·lectada per un **TaskMonitor**, associat a cada Worker. Quan el Worker executa una nova tasca, li associa un TaskMonitor, amb l'objectiu de monitoritzar la nova tasca en execució. Quan la tasca finalitza la seva execució, el Worker s'encarrega d'eliminar el seu TaskMonitor.

Com PeerMonitor, TaskMonitor compleix amb les funcionalitats d'un sensor. Recol·lecta la quantitat de CPU, memòria i espai de disc que esta usant la tasca, enviant un senyal quan es sobrepassi el llindar configurat per aquesta

tasca. A l'especificació XML d'execució de una tasca, l'aplicació que sol·licita l'execució de la tasca especifica, de forma opcional, els llindars d'us dels recursos computacionals per la tasca. En el cas de que una tasca no tingui especificat els llindars, el sistema assignarà uns llindars automàticament, acord al nombre de tasques executant-se i la càrrega de l'igual.

TaskMonitor utilitza els mateixos patrons de disseny que PeerMonitor: Composite i Observable. Mitjançant el patró Composite, es pot ampliar l'observació de les tasques, afegint nous sensors.

Els PeerMonitor i TaskMonitor aporten a LaCOLLA un mecanisme autònom de detecció de problemes en l'execució de les tasques. Si durant un estat de sobrecàrrega, l'EA rep una petició d'assignació de tasca a executar, aquesta serà rebutjada. Aquest mecanisme permet mantenir la integritat de l'igual sense sobrecarregar-lo, evitant l'ús de més recursos dels que ha aportat al grup.

Si es produeix una sobrecàrrega a un igual, el component PeerMonitor i/o TaskMonitor ho notifica al EA. L'EA rep el senyal dels sensors i inicia el procés de recuperació de l'igual, on es individualitza el problema de sobrecarrega, fins trobar la tasca o el conjunt de tasques que estan generant la sobrecàrrega.

El sistema distingeix entre dos tipus de sobrecàrrega: La produïda per l'execució de les tasques de LaCOLLA, i la produïda per tasques externes a LaCOLLA. Un exemple de sobrecàrrega externa és: l'usuari ha engegat un programari per multiplexar àudio i vídeo, què fa que l'ús de processador sigui molt intensiu. Llavors, quan es produeix un estat de sobrecàrrega, el sistema determina el tipus de sobrecàrrega, per aplicar el procés de recuperació adient. Aquest mecanisme de classificació de la sobrecàrrega ho proporciona l'EA, sumant els valors dels diferents TaskMonitor i comparant el resultat amb els llindars del PeerMonitor.

En cas que la sobrecàrrega estigui produïda per tasques externes a LaCOLLA, el sistema no suspèn cap tasca, ja que si ho fes, com a resultat, estarien totes les tasques suspeses i l'igual seguiria sobrecarregat, ja que les tasques que estan consumint massa recursos, no són del domini de LaCOLLA. Com que el sistema no és intrusiu en el sentit de no manipular cap procés o tasca que no sigui domini de LaCOLLA, el sistema no inicia cap procés de recuperació de tasques.

En cas que la sobrecàrrega estigui produïda per les tasques que té en execució l'EA, aquest suspèn la tasca o tasques que menys temps portin en execució, fins trencar la condició de sobrecarrega. Els motius pels quals el sistema suspèn les tasques que porten menys temps en execució són:

- No penalitzar les tasques que porten ja temps en execució. Es pretén minimitzar els casos en que s'ha de migrar una tasca, que potser ja estigui terminant la seva execució.

- Es suposa que abans de l'execució de aquestes tasques, l'igual no estava sobrecarregat. Aquesta hipòtesi es compleix en tots els casos en què la sobrecàrrega d'un igual esta provocada per l'execució de les tasques de LaCOLLA. No es compleix, però, en aquells casos en què la sobrecàrrega ha estat produïda per tasques externes a LaCOLLA.

Per cada tasca suspesa, l'EA envia el missatge **MsgChangedStateTask**, notificant a tots els TDA del grup, el nou estat de la tasca. A continuació, s'inicia el procés de recuperació de les tasques suspeses.

El procés de recuperació de les tasques que estan suspeses i que són les que provocaven la sobrecàrrega, consisteix en deixar passar un temps configurable i prudencial, per determinar l'acció a prendre amb les tasques. Llavors, es pot tenir els següents escenaris:

- Passat el temps, l'estat de l'igual és el mateix. L'EA té les mateixes tasques en execució. El sistema inicia el procés de migrar les tasques a un altre EA amb més disponibilitat.

- Passat el temps, l'estat ha canviat. Tasques que hi havien en execució ja han acabat i/o la càrrega de l'igual ha disminuït. En aquesta situació, el sistema va reanudant les tasques suspeses, sense carregar l'igual. Si totes les tasques no s'han pogut reanudar, les tasques restants són migrades a un altre EA amb més disponibilitat.

La decisió de deixar passar un temps des de la suspensió de les tasques, és per minimitzar els casos en que s'ha de migrar una tasca. La migració és un procés que penalitza les tasques que porten molt temps en execució i, que potser, ja estiguin terminant la seva execució, ja que al migrar la tasca a un altre EA, la tasca ha de començar de nou la seva execució.

Si les tasques són reanudades, l'EA envia el missatge **MsgChangedStateTask** per cadascuna de les tasques, als TDA, indicant el nou estat. Si per el contrari, cal iniciar el procés de migració, l'EA selecciona un dels TDA del grup, i li envia el missatge **MsgTasksNeedToSchedule**, adjuntant les tasques que estan suspeses, amb l'objectiu de iniciar el procés de migració.

Fins ara s'ha descrit com els EA obtenen la informació de càrrega dels iguals, PeerMonitor, i quin ús de recursos dels iguals, fan les tasques, mitjançant els TaskMonitor. També s'ha descrit, com els EA actuen davant una sobrecàrrega a un igual. A continuació es descriu com els TDA reben la informació de monitorització dels iguals, la qual és útil per determinar a quin EA se li pot assignar una tasca, basant-se en la seva informació de càrrega, així com per migrar les tasques a un altre igual amb més recursos computacionals.

Per replicar la informació de monitorització als TDA, s'utilitza el mecanisme actual dels TDA, per sincronitzar les característiques dels EA. S'afegeix al

missatge de les característiques d'un EA, la informació de càrrega de l'igual. Quan els TDA han de assignar una tasca a un EA, utilitzen la informació de càrrega que coneixen dels EA, per construir la llista de EA candidats, basant-se en la millor disponibilitat d'aquests respecte als requeriments de la tasca.

Quan els TDA reben el missatge MsgTasksNeedToSchedule, enviat pels EA indicant les tasques suspeses que provocaven la sobrecàrrega a l'igual on s'executaven, els TDA actualitzen el seu registre de tasques amb la informació de les tasques adjuntes al missatge i procedeixen a iniciar el procés de recuperació d'aquestes. Aquest procés de recuperació, de les tasques suspeses, consisteix en planificar la seva execució, cercant els TDA disponibles que puguin executar-les, sense sobrecarregar l'igual on s'executen. Les tasques es van migrant per prioritat de les tasques, i les que no es puguin migrar són encolades, a l'espera de recursos computacionals al grup.

La migració d'una tasca, per part d'un TDA, consisteix en negociar, de nou, la seva execució. Per aquest motiu, es seleccionen els EA connectats al grup, amb major disponibilitat, i s'inicia el procés de negociació amb els EA. Si no hi ha EA disponibles (no hi ha suficients recursos computacionals al grup per poder executar la tasca, sense sobrecarregar algun igual), la tasca queda pendent d'assignació, fins que hi hagin EA disponibles per la seva execució.

Cal destacar, que en una situació en la que només hi ha un EA connectat al grup, i aquest ha suspès una tasca que donava problemes al igual, no tornarà a acceptar l'execució de la tasca fins que hi hagi passat un temps predeterminat. D'aquesta manera, el sistema evita que hi hagin cicles recurrents.

A aquesta nova arquitectura, s'ha afegit el concepte de prioritat de un tasca, amb l'objectiu de poder prioritzar l'execució d'una determinada tasca davant d'una altre, en el mateix igual. El sistema defineix les següents prioritats: Màxima, Normal i Mínima. Per defecte, si no s'especifica la prioritat,a l'especificació XML d'execució de la tasca, el sistema li assignarà, automàticament, prioritat Normal**.** El Worker és l'encarregat d'establir la prioritat de la tasca que va a executar, o s'està executant. L'EA és l'encarregat de canviar la prioritat a una tasca que s'està executant. Aquesta decisió la prendrà, autònomament, durant la recuperació de l'igual, en cas de sobrecàrrega.

Pel que fa a les prioritats i els llindars del recursos s'especifiquen mitjançant l'element requirement definit al DTD que defineix la tasca.

SERVEIS DE CONSULTES

Aquests serveis proporcionen informació sobre les tasques que s'estan executant, així com de la càrrega dels iguals connectats a un determinat grup. Aquesta informació es fa accessible a les aplicacions de LaCOLLA. Per exemple, una aplicació de monitorització que mostri l'estat de les tasques que s'estan executant, els recursos que aquestes consumeixen, etc...

A continuació es descriu el disseny de cada funcionalitat dels serveis de consulta.

CAS D'US: OBTENIR LES TASQUES EN EXECUCIÓ D'UN GRUP

Descripció	Servei que proporciona les tasques en execució o pendents d'execució d'un determinat grup de LaCOLLA.
Actors implicats	Aplicació de LaCOLLA

Les aplicacions instancien un objecte de la classe ApiImpl que implementa la interfície Api que ofereix LaCOLLA. Per obtenir una llista de totes les tasques que s'estan executant a un determinat grup, l'aplicació invoca el mètode getTaskList de ApiImpl, indicant l'identificador del grup al que pertany.

L'objecte ApiImpl instancia un objecte doGetTaskList indicant els paràmetres rebuts i afegint al membre i la instancia de l'UA, que dona servei a l'aplicació que va invocar aquest servei, i s'invoca al mètode getTaskList de l'UA. L'UA selecciona un TDA que estigui connectat al grup indicat com a paràmetre, i envia el missatge msgGetTaskList, retornant el control a l'aplicació. La instància de la classe msgServiceTDA del TDA seleccionat, rep el missatge i invoca al mètode doGetTaskLists del TDA.

El TDA envia als UA un missatge msgTaskList, indicant les tasques que el TDA té registrades. En aquest missatge s'adjunta un vector amb els identificador de les tasques que te registrat el TDA.

Els UA reben el missatge mitjançant la classe msgServiceUA, que executa el mètode doTaskList, que, si es tracta del membre que va realitzar la petició, proveeix la informació a les aplicacions, invocant el mètode notifyTaskList de la classe ApplicationSideApi, que obligatòriament les aplicacions han d'implementar.

CAS D'US: OBTENIR INFORMACIÓ DE CARREGA D'UNA TASCA

Descripció	Servei que proporciona informació de monitorització de una determinada tasca.
Actors implicats	Aplicació de LaCOLLA

Les aplicacions instancien un objecte de la classe ApiImpl que implementa la interfície Api que ofereix LaCOLLA. Per obtenir la informació de càrrega d'una determinada tasca, l'aplicació invoca el mètode getTaskMonitorInfo de ApiImpl, indicant l'identificador de la tasca i el grup al que pertany.

L'objecte ApiImpl instancia un objecte doGetTaskMonitorInfo indicant els paràmetres rebuts i afegint el membre i la instancia de l'UA, que dona servei a l'aplicació que va invocar aquest servei,i s'invoca al mètode getTaskMonitorInfo de l'UA. L'UA busca un TDA que estigui connectat al grup indicat com paràmetre i envia el missatge msgGetTaskMonitorInfo, retornant el control a l'aplicació. La instancia de la classe msgServiceTDA del TDA seleccionat, rep el missatge i invoca al mètode doGetTaskMonitorInfo del TDA.

Si el TDA seleccionat no conté informació de la tasca especificada, encamina la petició a la resta de TDA del grup.

El TDA que tingui registrada la tasca especificada a la sol·licitud, envia al EA que l'executa, el missatge msgGetTaskMonitorInfo. La instancia de la classe msgServiceEA de l'EA que executa la tasca, rep el missatge i invoca al mètode doGetTaskMonitorInfo de l'EA. L'EA envia a tots els UA connectats al grup, el missatge msgTaskInfoMonitor, indicant la informació de càrrega de la tasca. Al missatge s'adjunta una instancia de la classe InfoMonitor, que conté la informació de càrrega dels recursos que consumeix la tasca. InfoMonitor conté la següent informació: ús de CPU, memòria i espai de disc assignat.

Els UA reben el missatge mitjançant la classe msgServiceUA, que executa el mètode doTaskMonitorInfo, que, si es tracta del membre que va realitzar la petició, proveeix la informació a les aplicacions, invocant el mètode notifyTaskMonitorInfo de la classe ApplicationSideApi, que obligatòriament les aplicacions han d'implementar.

CAS D'US: OBTENIR INFORMACIÓ DE CARREGA DELS IGUALS

Descripció	Servei que proporciona informació de carrega dels iguals d'un determinat grup.
Actors implicats	Aplicació de LaCOLLA

Les aplicacions instancien un objecte de la classe ApiImpl que implementa la interfície Api que ofereix LaCOLLA. Per obtenir la informació de càrrega de tots els iguals connectats a un determinat grup, l'aplicació invoca al mètode getPeerMonitorInfo de ApiImpl, indicant l'identificador del grup del qual vol obtenir la informació.

L'objecte ApiImpl instancia un objecte doGetPeerMonitorInfo indicant els paràmetres rebuts i afegint el membre i la instancia de l'UA, que dona servei a l'aplicació que va invocar aquest servei, i invoca el mètode getPeerMonitorInfo de l'UA. L'UA envia el missatge msgGetPeerMonitorInfo, a tots els EA que estan connectats al grup, i retorna el control a l'aplicació. La instancia de la classe msgServiceEA dels EA, rep el missatge i invoca al mètode doGetPeerMonitorInfo de l'EA.

Els EA envien als UA el missatge msgPeerInfoMonitor, indicant la informació de càrrega. Al missatge s'adjunta una instancia de la classe InfoMonitor, que conté la informació de càrrega de l'igual.

Els UA reben el missatge mitjançant la classe msgServiceUA, que executa el mètode doPeerMonitorInfo, que, si es tracta del membre que va realitzar la petició, proveeix la informació a les aplicacions, invocant el mètode notifyPeerMonitorInfo de la classe ApplicationSideApi, que obligatòriament les aplicacions han d'implementar.

CAPÍTOL 4 – VALIDACIÓ

En aquest capítol es validen els components i els mecanismes interns de l'arquitectura proposada. Aquesta validació s'ha fet verificant que es compleixen les funcionalitats que aquest treball aporta:

- Garantir la integritat de l'igual que executa les tasques de LaCOLLA, i no consumir més recursos dels que l'igual aporta al grup.
- Detecció de problemes de sobrecàrrega als iguals.
- Tractament de les tasques en execució als iguals amb sobrecàrrega.

La validació s'ha fet, implementant un prototipus d'aquesta arquitectura. Aquest prototipus es detalla a continuació.

IMPLEMENTACIÓ DEL PROTOTIPUS

Per poder validar aquesta arquitectura, s'ha construït un prototipus que avalua diferents escenaris, comprovant com es comporta el sistema amb situacions de sobrecàrrega,i que al final, és compleixin les funcionalitats aportades per aquest treball.

La implementació d'aquest prototipus esta basada en una versió estable de LaCOLLA, amb la funcionalitat d'execució de tasques distribuïdes, aportada pel treball d'Antoni Ribes.

Algunes característiques d'aquesta versió, i que per coherència s'han mantingut en el prototipus d'aquest treball, són les següents:

- Com a llenguatge de programació s'ha pres el Java perquè facilita la programació OOP (orientada a objectes) i perquè el prototipus anterior ja estava fet en Java.
- Per a la comunicació entre aplicacions i LaCOLLA s'ha usat RMI que és l'eina que ofereix Java per a realitzar crides remotes. S'ha triat aquesta eina com una solució temporal perquè és la que permetia poder començar a treballar amb el prototipus més ràpidament, malgrat que va en contra de l'objectiu d'independència de programari de LaCOLLA. En un futur s'implementarà un mecanisme que garantirà aquesta independència de la plataforma i el programari que usin les aplicacions.

- Per a la comunicació entre components de LaCOLLA s'ha utilitzat el pas de missatges a través de sockets, que sí que permet garantir la independència de plataforma entre membres.

Per la construcció del prototipus s'han utilitzat els models proposats a l'enginyeria del programari i tècniques de desenvolupament de programari, estudiades a les assignatures de Enginyeria del programari i Tècniques de Desenvolupament del Programari de la UOC.

A continuació, es fa una descripció de les responsabilitats de cadascuna de les classes que s'han desenvolupat, agrupades segons el package Java al que pertanyen.

PACKAGE LACOLLA.API

En aquest paquet es troba la definició (Api) i implementació (ApiImpl) de la interfície que ofereix LaCOLLA a les aplicacions. A aquesta interfície ha calgut incloure-li les funcions referents als serveis de consulta: getTaskList, getTaskMonitorInfo, getPeerMonitorInfo.
També trobem la definició de la interfície que han d'implementar obligatòriament les aplicacions per a usar LaCOLLA (ApplicationsSideApi), on s'han inclòs les notificacions que poden rebre per l'ús dels serveis de consulta: notifyPeerMonitorInfo, notifyTaskMonitorInfo i notifyTaskList.

PACKAGE LACOLLA.CORE.COMPONENTS

En aquest paquet es troba la implementació dels components agents TDA,EA i UA. S'han modificat tots tres, per afegir-hi les noves funcionalitats.

PACKAGE LACOLLA.CORE.MESSAGES

Cadascun dels nous missatges que s'envien els components TDA, EA i UA s'implementa en forma de classe, i s'agrupen en aquest paquet.

PACKAGE LACOLLA.CORE.TASK

En aquest paquet es troba la implementació de la funcionalitat d'execució de tasques, ja existent. S'han afegit les classes PeerMonitor, TaskMonitor, InfoMonitor i Worker i s'han modificat les classes ja existents Task i javaWorker.

PACKAGE LACOLLA.CORE.UTIL

Aquest paquet el composen classes de diverses finalitats, que donen servei a la resta de paquets. S'han modificat les classes: msgServiceEA, msgServiceTDA i msgServiceUA, que són les responsables de rebre els missatges per als EA,TDA i UA.
Dins el subpaquet runnable, s'han afegit les classes doGetPeerMonitorInfo, doGetTaskMonitorInfo i doGetTaskList, que duen el control del procés engegat per la petició d'una aplicació.

Els sensors de CPU, memòria i disc s'han implementat mitjançant C++ ja que Java, per raons obvies, no permet accedir a aquesta informació. Per aquest motiu s'ha fet servir el Java Native Interface (JNI), per delegar aquesta part de la implementació a un llenguatge que ho permetés. La implementació realitzada, per motius logístics, és sobre Win32. Per recol·lectar informació de monitorització del sistema s'ha empleat el Performance API del Win32 SDK.

VALIDACIÓ DE LES FUNCIONALITATS

S'han confeccionat uns jocs de prova, per tal de validar les funcionalitats, que aquest treball, aporta a LaCOLLA.

VALIDACIÓ DE LA INTEGRITAT DE L'IGUAL

Aquesta funcionalitat, garanteix la integritat de l'igual que executa les tasques de LaCOLLA, evitant que es consumeixin més recursos dels que l'igual aporta al grup. Per a validar aquesta funcionalitat s'ha dissenyat un joc de proves que es detalla a continuació.

S'ha preparat una màquina amb la següent configuració de components:

- Un GAPA, un RA i un UA
- Un TDA: el temporitzador que engega el mètode doServiceTimer, encarregat entre altres coses de la sincronització de característiques dels EA, s'ha programat a 5 segons.
- Un EA: Aquest EA s'ha configurat amb els llindars de CPU molt baixos, per assegurar,directament, sobrecàrrega a l'igual: CPU 0%, memòria 50% i disc 50%.

S'ha llançat una petició d'execució d'una tasca i la tasca s'ha quedat com pendent d'execució. L'únic EA connectat al grup, no accepta l'execució de la tasca, ja que es troba en una situació de sobrecàrrega.

A continuació s'ha preparat un EA amb uns llindars de recursos més normals: CPU 50%, memòria 50%, disc 50%. La tasca que està en estat pendent, es assignada a l'últim EA connectat i comença la seva execució.

D'aquest joc de proves se'n dedueix que el sistema no permet executar tasques als iguals que estan en condició de sobrecàrrega,i es garanteix que no s'utilitzen més recursos dels que ha aportat l'igual.

VALIDACIÓ DE LA DETECCIÓ DE PROBLEMES I MECANISME DE RECUPERACIO

Aquesta funcionalitat determina si les tasques associades a un EA estant tenint problemes amb la seva execució. Mitjançant el sistema de monitorització es pot determinar si les tasques a un igual estant produint sobrecàrrega, o si per altres causes l'igual està a aquesta situació.

S'ha preparat una màquina amb la següent configuració de components:

- Un GAPA, un RA i un UA
- Un TDA: el temporitzador que engega el mètode doServiceTimer, encarregat entre altres coses de la sincronització de característiques dels EA, s'ha programat a 5 segons.
- Un EA: Aquest EA s'ha configurat amb els llindars de CPU 50%, memòria 50% i disc 50%. El temporitzador per no acceptar tasques que han estat suspeses per aquest EA, s'ha posat a 2 minuts.

S'han sotmès dos tasques al grup. Una versió original de la tasca "bucle" i una versió modificable d'aquesta, que no fa sleep, i per tant, consumeix molta CPU.

La primera tasca no sobrecàrrega l'igual. Quan s'ha començat a executar la darrera tasca, aquesta ha sobrecarregat l'igual. El sistema ho ha detectat i ha iniciat el procés de recuperació de l'igual. Per fer-ho, ha suspès l'execució de la darrera tasca, la que ha fet sobrecarregar l'igual. Una vegada suspesa, s'ha trencat la condició de sobrecàrrega. L'execució de la tasca ha quedat suspesa, i l'igual s'ha recuperat de la sobrecàrrega. En aquest cas, com que només hi ha connectat el mateix EA que ha suspès la tasca, l'EA no acceptarà l'execució de la tasca fins que hi hagin passat els 2 minuts, marcats pel temporitzador.

VALIDACIÓ DELS SERVEIS DE CONSULTA

S'ha implementat una aplicació, per invocar els serveis de consulta: Obtenir la llista de tasques en execució, Obtenir càrrega dels iguals i Obtenir informació de monitorització d'una tasca. Mitjançant aquesta aplicació s'ha pogut comprovar el correcte funcionament d'aquests serveis, així com validar les modificacions fetes als components existents.

CAPÍTOL 5 – CONCLUSIONS I TREBALLS FUTURS

L'objectiu d'aquest treball ha estat estendre l'oferta de funcionalitats de LaCOLLA, definint uns mecanismes que permetin detectar problemes en les tasques que LaCOLLA executa, i el tractament de les tasques en execució als iguals amb sobrecàrrega.

En aquest treball es presenta l'arquitectura proposada per ampliar LaCOLLA amb aquestes funcionalitats. S'han definit els components i els mecanismes interns necessaris, que permeten monitoritzar l'estat dels iguals que executen les tasques, així com l'estat de les tasques que estan en execució. Aquesta informació permès determinar si un determinat igual està tenint problemes i recuperar aquests iguals i les tasques que tenien en execució.

Per a validar aquesta arquitectura s'ha construït un prototipus en el que s'han implementat els mecanismes necessaris per a poder simular diverses situacions, podent comprovar que:

- Es garanteix la integritat dels iguals que ofereixen els seus recursos computacionals, per executar tasques.

- Es garanteix que una tasca a un igual amb problemes, continuarà, si cal, la seva execució en un altre igual.

- El sistema és robust gràcies a la replicació de responsabilitats, que permet garantir que el servei sempre està actiu malgrat les possibles desconnexions i connexions dels membres del grup.

- El sistema és fiable gràcies al sincronisme virtual de LaCOLLA i també a la replicació de responsabilitats, que permeten conèixer l'estat del sistema d'una manera que es percep com immediata i actuar en conseqüència, així com garantir que no es perd informació.

Com a treball pendent i de futur queda implementar la resta de mecanismes definits que no s'han inclòs al prototipus, i comprovar que són vàlids o proposar-ne d'alternatius, així com aprofundir en la planificació de les tasques en situacions de sobrecàrrega o estudiar un mecanisme més òptim per migrar una tasca a un altre igual, sense haver de començar de nou la seva execució des d'un principi.

Al prototipus s'han desenvolupat les funcionalitats necessàries per a complir amb els objectius de la validació, però hi ha d'altres que han quedat pendents d'implementar, com l'assignació automàtica de prioritats i la migració de les tasques per ordre de prioritat. Aquestes tasques no han estat implementades al prototipus, per que no s'han pogut validar amb l'entorn del que disposo. Només tinc un ordinador, i trobar un grup de ordinadors a Internet per poder validar aquestes característiques és difícil.

Actualment la migració d'una tasca, significa començar-la de nou en un altre igual, i això és una limitació en el treball actual. Aquest mecanisme perjudica les tasques que porten temps executant-se, i s'han de migrar. En treballs futurs, es podria implementar un mecanisme de migració que permeti executar la tasca a l'altre igual, des de el punt de suspensió, tal i com ho fan altres plataformes com Condor.

Un altre característica, que podria incrementar l'eficiència d'aquest sistema, seria definir una interfície amb les tasques, de tal manera que aquestes vagin informant del percentatge de treball realitzat. D'aquesta manera, amb aquesta dada, el sistema podria decidir no migrar-la (ja que per exemple té el 90% de la seva feina feta) i aplicar altres politiques a l'igual.

La planificació de tasques és un tema que també es podria aprofundir en un treball futur. Es podria dissenyar politiques de planificació més complexes, i segurament més efectives que les actuals.

BIBLIOGRAFIA

Marques, 2003: LaCOLLA: una infraestructura autònoma i autoorganitzada per facilitar la col·laboració.

Ribes, 2004: Una extensió de LaCOLLA: abstracció de processament.

Condor: High Throughput Computing, http://www.cs.wisc.edu/condor/

Enginyeria del Programari I: Benet Campderrich i Falgueras, UOC