Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Distribuzione ed esecuzione automatica di task in griglie pervasive

G. De Pietro, A. Coronato, L.Gallo

RT-ICAR-NA-2006-15 **08-2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Distribuzione ed esecuzione automatica di task in griglie pervasive

G. De Pietro[1], A. Coronato[2], L. Gallo[1]

Rapporto Tecnico N.: **RT-ICAR-NA-2006-15**

Data: **08-2006**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli
[2] SASIT-CNR, Via P. Castellino 111, 80131 Napoli

Distribuzione ed esecuzione automatica di task in griglie pervasive

Antonio Coronato[2], Giuseppe De Pietro[1], Luigi Gallo[1]

[1]ICAR-CNR, Via Castellino 111, 80131 Napoli, Italia
{depietro.g, gallo.l@na.icar.cnr.it}
[2]SASIT-CNR, Via Castellino 111, 80131 Napoli, Italia
{coronato.a@na.drr.cnr.it}

Abstract

La congiunzione dei paradigmi di Grid computing e Pervasive computing ha portato alla generazione del paradigma di *Pervasive Grid Computing*, il cui scopo è quello di estendere le griglie classiche con caratteristiche di pervasività quali l'integrazione spontanea e trasparente di dispositivi mobili, context-awareness, pro-activity, e così via.

In questo technical report viene presentata una infrastruttura software per integrare dispositivi mobili come risorse attive in una griglia. In particolare è stato sviluppato un servizio capace di distribuire ed eseguire dinamicamente task utente su una griglia di dispositivi fissi e mobili. Tali dispositivi partecipano alla griglia in maniera completamente trasparente ai loro possessori, la potenza computazionale viene fornita, a chi ne faccia richiesta, senza alcuna operazione di configurazione ed in maniera affidabile. Gli utenti che vogliono eseguire applicazioni sulla griglia devono semplicemente sottomettere il loro codice all'ambiente.

1. Introduzione

Una "griglia" nasce allo scopo di permettere una condivisione flessibile, sicura e coordinata di risorse tra gruppi dinamici di individui, istituzioni e sistemi. L'attributo

“pervasivo” indica una combinazione di elevata mobilità ed integrazione nell’ambiente: il software è pervasivo se è in grado di adattarsi ai dispositivi che via via si rendono disponibili. Dunque una griglia pervasiva può essere vista come una rete dinamica di dispositivi eterogenei per caratteristiche e per mobilità, che permette una condivisione flessibile e coordinata di risorse di vario tipo.

I sistemi a griglia tradizionali sono software di notevoli dimensioni che permettono di prenotare, riservare, monitorare lo stato di risorse eterogenee, ma mal si adattano all’integrazione di dispositivi mobili, sia per la connettività che in questi ultimi è spesso intermittente, sia per il tipo di applicazione, troppo estesa per le limitate risorse degli stessi. Allo stato attuale tali sistemi vengono per lo più utilizzati per interconnettere tra loro grossi centri di calcolo.

Se però consideriamo la legge di Metcalfe: "Il valore di una rete è proporzionale approssimativamente al quadrato del numero dei suoi nodi" e se consideriamo che i dispositivi divengono con il passare degli anni sempre più piccoli e più veloci, si capisce che l’estensione delle griglie ai dispositivi mobili è indispensabile, e deve essere fatta in maniera pervasiva, dispensando quindi l’utente dall’esecuzione di complesse operazioni di installazione e manutenzione.

Il risultato al quale si è tesi è quello di utilizzare risorse informatiche con la stessa facilità della corrente elettrica o dell'acqua, connettendosi ad una rete della quale in teoria non si deve conoscere nulla, un approccio questo recentemente denominato “Utility Computing”.

Il lavoro di tesi è stato quindi finalizzato alla creazione di un modello e di un servizio su di esso basato, per consentire l’integrazione “user-friendly” di un gruppo dinamico di dispositivi eterogenei allo scopo di condividere la potenza di calcolo degli stessi. Creare quindi una griglia pervasiva “light”, in cui è il software ad adattarsi al dispositivo e nella quale tutto quello che si richiede all’utilizzatore è di effettuare una iscrizione alla griglia.

L’idea alla base del modello è stata quella di rendere il software mobile tanto quanto i dispositivi che compongono la griglia stessa: l’intera struttura software deve essere composta da “agenti mobili”, evoluzioni degli oggetti tradizionali in quanto ogni agente costituisce un codice attivo, in grado cioè di intraprendere iniziative e di agire autonomamente. In tal modo si può inviare sul dispositivo solo la porzione di

software realmente necessaria. Nel modello proposto qualunque dispositivo dotato di una connessione internet può entrare a far parte della griglia, e condividere la sua potenza di calcolo con gli altri dispositivi coinvolti.

In un contesto di dispositivi mobili e quindi con connessioni internet inaffidabili, batterie di durata limitata, e con utenti che in ogni momento possono decidere di staccarsi dalla griglia, l'affidabilità diventa un aspetto cruciale. Per garantirla il modello suggerisce un meccanismo di clonazione: ogni agente ha un clone pronto a prendere il suo posto se per qualsiasi motivo il primo dovesse smarrirsi.

Ogni volta che c'è necessita di distribuire un task complesso in un certo numero di task che non necessitano collaborazione tra loro, o se c'è necessita di eseguire un singolo task ma con diversi valori in ingresso, questo modello risulta particolarmente indicato.

Il servizio, che è stato denominato *DDT (Dynamic Distribution and execution of Tasks)* è stato realizzato interamente in Java, proprio per favorirne la portabilità, ed utilizza *JADE (Java Agent DEvelopment Framework)* per implementare il paradigma della programmazione ad agenti.

2. Il servizio di distribuzione ed esecuzione automatica dei task

2.1 Caratteristiche principali

L'infrastruttura software proposta consente agli utenti di una griglia di distribuire ed eseguire task su un gruppo dinamico di dispositivi fissi e mobili.

Rispetto alle griglie classiche gli elementi di innovazione sono:

a. **Integrazione trasparente di dispositivi mobili come risorse attive** – Per questa funzionalità si richiede solo l'installazione sul dispositivo mobile di un piccolo software plug-in che consiste in un contenitore di agenti. Fatto ciò, ogni volta che il dispositivo entrerà nell'ambiente, diverrà una risorsa attiva; questo significa che l'ambiente potrà allocare ed eseguire task su di esso in modo completamente trasparente al suo possessore;

b. **Esecuzione diretta di applicazioni** – Gli utenti devono sottomettere solo il loro codice ed alcuni parametri d'esecuzione. Al contrario, nelle griglie classiche gli utenti devono prima richiedere delle risorse alla griglia, poi lanciare i task, controllarli, prelevare i risultati e quindi rilasciare le risorse utilizzate. Richiedere solo la sottomissione del codice agli utenti, invece, da una parte semplifica il lancio di task sulla griglia, dall'altra protegge la griglia stessa da utenti maliziosi o inesperti;

c. **Esecuzione affidabile dei task** – I task vengono incapsulati in agenti mobili che possono essere allocati dall'ambiente sia su dispositivi fissi sia su mobili. Grazie ad un meccanismo di clonazione per gli agenti mobili e ad un protocollo di checkpointing per i dispositivi ed i task, l'ambiente può gestire in maniera affidabile la caduta improvvisa di uno o più dispositivi della griglia.

2.2 Architettura

L'architettura consiste dei seguenti componenti hardware/software:

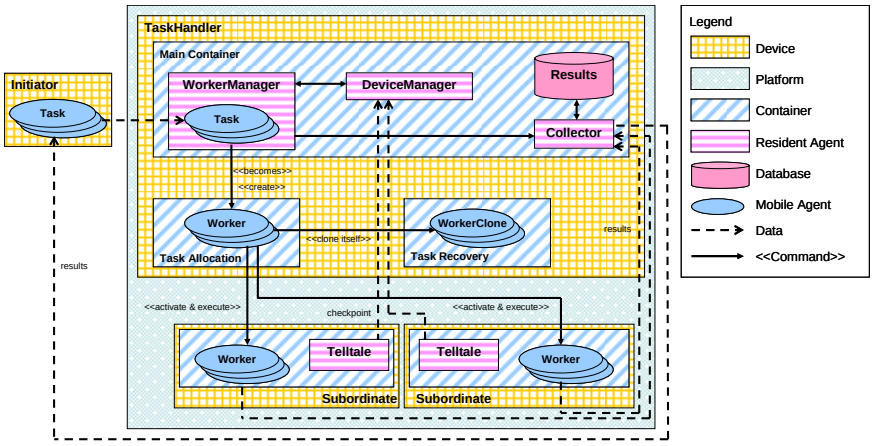

Figura 1 – Architettura

- ***Container*** – E' l'ambiente di run-time che fornisce le funzionalità di base per eseguire e coordinare gli agenti mobili;
- ***Platform*** – E' l'insieme dei nodi della griglia su cui è presente un *Container*, in altre parole è la griglia vera e propria;

- ***TaskHandler*** – E' il componente hardware che ospita i componenti incaricati della coordinazione degli agenti e della gestione della griglia. L'utente sottomette ad esso i task da eseguire;
- ***Initiator*** – E' il componente hardware su cui sono inizialmente presenti i task da distribuire ed eseguire sulla griglia. *Initiator* e *TaskHandler* possono anche essere lo stesso dispositivo;
- ***Subordinate*** – E' il componente hardware che ospita gli agenti mobili durante la loro esecuzione. Può essere un dispositivo tanto fisso quanto mobile;
- ***Worker*** – E' l'agente che incapsula il task utente. Il suo compito consiste nel migrare su un *Subordinate*, eseguire il task utente, quindi inviare i risultati al *Collector*. Più *Worker* possono essere inviati su uno stesso *Subordinate*;
- ***Telltale*** – E' l'elemento software incaricato di monitorare alcuni parametri del *Subordinate* su cui si trova. Le decisioni in merito all'allocazione ed alla conseguente migrazione dei *Worker* (e quindi dei task utente) vengono prese sulla base di tali parametri;
- ***DeviceManager*** – Questo agente, che risiede solo sul dispositivo *TaskHandler*, riceve dei checkpoint dai *Telltale* e gestisce la lista di tutti i *Subordinate* attivi nella griglia. Suo compito è anche quello di individuare la caduta di dispositivi della griglia, in modo da avviare procedure di recupero dei task utente ivi presenti;
- ***WorkerManager*** – E' l'agente che coordina la migrazione dei *Worker* nell'ambiente basandosi sulle informazioni fornitegli dal *DeviceManager*. Risiede unicamente sul dispositivo *TaskHandler*;
- ***Collector*** – Questo agente, anch'esso presente solo sul dispositivo *TaskHandler*, riceve i risultati dagli agenti attivi e li immagazzina nell'apposito archivio *Results*.

Il compito di gestione della griglia è dunque ripartito tra gli agenti *DeviceManager* e *WorkerManager*: il primo conosce tutto sui dispositivi coinvolti; il secondo conosce tutto sui *Worker* e sui loro task.

2.3 Funzionamento

Ogni volta che un utente vuole eseguire un'applicazione, deve sottometterla al dispositivo *TaskHandler*, o far divenire il suo dispositivo un *TaskHandler* lanciando una piccola applicazione Java. Fatto ciò, il *TaskHandler* incapsulerà i task ricevuti in *Worker* e li metterà nel *Container TaskAllocation*, dove resteranno in attesa di essere lanciati su un dispositivo della griglia. Una volta schedulato, ma prima di lasciare il *TaskHandler*, il *Worker* deve clonarsi nel *Container TaskRecovery*; fatto ciò, il task viene allocato su un dispositivo *Subordinate*. L'allocazione viene guidata dal *DeviceManager* che dispone di informazioni sullo stato dei dispositivi facenti parte della griglia. A questo punto possono verificarsi due possibili scenari:

1 - il *Worker* completa la sua esecuzione, invia i risultati al *Collector* che li memorizza nell'archivio *Results*;

2 – il *Worker* non riesce a completare il suo task, questo fallimento viene individuato dal *DeviceManager*, il quale non riceve i checkpoint dal *Telltale* del *Subordinate* su cui è in esecuzione. A questo punto il *DeviceManager* attiva il clone precedentemente creato e lo fa migrare nel *Container TaskAllocation*, dove resterà in attesa di migrare, ma non prima di clonarsi nuovamente, su un nuovo *Subordinate*.

Questo meccanismo non consente di individuare fallimenti la cui causa è imputabile al task incapsulato nel *Worker*, in quanto l'individuazione è basata sui checkpoint inviati dall'agente *Telltale* residente sul dispositivo, e questi non conosce l'esito dell'esecuzione ma verifica solo che il dispositivo sia collegato alla griglia.

Al contrario, fallimenti dovuti a disconnessioni di *Subordinate* per scelta del loro possessore, o per mancanza improvvisa di connettività (ad esempio uscita dalla zona di copertura Wi-Fi), o per spegnimento del dispositivo (ad esempio esaurimento della batteria di un dispositivo mobile), vengono invece correttamente gestiti grazie al meccanismo della clonazione.

3. Conclusioni

In questo technical report sono stati proposti un modello ed un servizio per consentire la distribuzione ed esecuzione dinamica di task, sottomessi da un utente, su

di una griglia di dispositivi fissi e mobili, in maniera pervasiva. Seguendo il paradigma dell'Utility Computing, la potenza computazionale viene fornita in modo completamente trasparente all'utente.

L'approccio attuale è applicabile ogni volta che l'applicazione da eseguire consiste o in un singolo task che deve essere eseguito molte volte (anche con differenti valori in ingresso) o in molti differenti task che non necessitano cooperazione tra loro.

I lavori futuri consisteranno nello sviluppare modelli di coordinazione tali da consentire una cooperazione inter-task, nuovi meccanismi atti a garantire maggiore affidabilità e sicurezza, nel migliorare l'algoritmo di scheduling.

4. Bibliografia

[1] http://www.utilitycomputing.com

[2] I. Foster, C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure". Morgan Kaufmann, 1999.

[3] D. Saha and A. Murkrjee, "Pervasive Computing: A Paradigm for the 21st Century", IEEE Computer, March 2003.

[4] B. Clarke and M. Humphrey, "Beyond the 'Device as Portal': Meeting the Requirements of Wireless and Mobile Devices in the Legion of Grid Computing System", International Parallel and Distributed Processing Symposium, IPDPS 2002.

[5] T. Phan, L. Huang and C. Dulan, "Challenge: Integrating Mobile Devices Into the Computational Grid", International Conference on Mobile Computing and Networking, MobiCom 2002.

[6] N. Daves, A. Friday, and O. Storz, "Exploring the Grid's Potential for Ubiquitous Computing", IEEE Pervasive Computing, April-June 2004.

[7] V. Hingne, A. Joshi, T. Finin, H. Kargupta, E. Houstis, "Towards a Pervasive Grid", International Parallel and Distributed Processing Symposium, IPDPS 2003.

[8] G. Coulson, P. Grace and G. Blair, D. Duce, C. Cooper and M. Sagar, “A Middleware Approach for Pervasive Grid Environments”, Workshop on Ubiquitous Computing and e-ResearchNational eScience Centre, Edinburgh, UK 18-19 May 2005.

[9] C. F. R. Geyer et. al., “GRAPEp: Towards Pervasive Grid Executions”, III Workshop on Computational Grids and Applications, WCGA 2005.

[10] http://www.globus.org

[11] I. Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems”, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005, also available on-line at: www.globus.org.

[12] F. Bellifemmine, A. Poggi and G. Rimassa, “Jade Programmers Guide”, http://sharon.cselt.it/projects/jade

[13] M. Ciampi, A. Coronato, G. De Pietro, and M. Esposito, “Handling Structured Classes of Events in Pervasive Grids”, to appear in the proc. of the 4th International Metainformatics Symposium, MIS 2005, as Lecture Notes in Computer Science, LNCS, Springer Verlag.

[14] A. Coronato, M. Ciampi, G. De Pietro, and M. Esposito, “A Location Service for Pervasive Grids”, in the proc. of the International Conference on Systems, Computing Sciences and Software Engineering (SCS2 2005).

[15] F. Bellifemmine, A. Poggi, and G. Rimassa, “JADE – FIPA Compliant Agent Framework”, In Proceedings of PAAM, 1999