Scalable Computational Science

Carmine Spagnuolo

Università degli Studi di Salerno

Dipartimento di Informatica

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione

DOCTOR OF PHILOSOPHY

Computer Science

Parallel and Distributed Computing

Scalable Computational Science

Carmine Spagnuolo

Supervisor Prof. Vittorio Scarano

Supervisor Dott. Gennaro Cordasco

2017

Carmine Spagnuolo
Scalable Computational Science
Parallel and Distributed Computing, Supervisors: Prof. Vittorio Scarano and Dott. Gennaro Cordasco

Università degli Studi di Salerno

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione
Dipartimento di Informatica
Via Giovanni Paolo II, 132
84084 , Salerno

Abstract

Computational science anche conosciuta come calcolo scientifico è un settore in rapida crescita che usa il calcolo avanzato per affrontare problemi complessi. Questa nuova disciplina, combina tecnologia, moderni metodi computazionali e simulazioni per affrontare problemi troppo difficili da poter essere studiati solo in teoria o troppo pericolosi e costosi per poter essere riprodotti sperimentalmente in laboratorio.

I progressi dell'ultimo ventennio in computational science hanno sfruttato il supercalcolo per migliorare le performance delle soluzioni e permettere la crescita dei modelli, in termini di dimensioni e qualità dei risultati ottenuti. Le soluzioni adottate si avvalgono del calcolo distribuito: è ben noto che la velocità di un computer con un singolo processore sta raggiungendo dei limiti fisici. Per queste ragioni, la computazione parallela e distribuita è diventata il principale paradigma di calcolo per affrontare i problemi nell'ambito della computational science, in cui la scalabilità delle soluzioni costituisce la sfida da affrontare.

In questa tesi vengono discusse la progettazione e l'implementazione di *Framework*, *Linguaggi Paralleli* e *Architetture* che consentono di migliorare lo stato dell'arte della *Scalable Computational Science*. In particolare, i maggiori contributi riguardano:

Frameworks. La proposta di D-MASON, una versione distribuita di MASON, un toolkit Java per la scrittura e l'esecuzione di simulazioni basate su agenti (Agent-Based Simulations, ABSs). D-MASON introduce la parallelizzazione a livello framework per far si che gli scienziati che lo utilizzano (ad esempio un esperto con limitata conoscenza della programmazione distribuita) possano rendersi conto solo minimamente di lavorare in ambiente distribuito (ad esempio esperti del dominio con limitata esperienza o nessuna esperienza nel calcolo distribuito). D-MASON è un progetto iniziato nel 2011, il cui principale obiettivo è quello di superare i limiti del calcolo sequenziale di MASON, sfruttando il calcolo distribuito. D-MASON permette di simulare modelli molto più complessi (in termini di numero di agenti e complessità dei comportamenti dei singoli agenti) rispetto a MASON e inoltre consente, a parità di calcolo, di ridurre il tempo necessario ad eseguire le simulazioni MASON. D-MASON è stato progettato in modo da permettere la migrazione di simulazioni scritte in MASON con un numero limitato di modifiche da apportare al codice, al fine di garantire il massimo della semplicità d'uso.

D-MASON è basato sul paradigma Master-Worker, inizialmente pensato per sistemi di calcolo eterogenei, nelle sue ultime versioni consente l'esecuzione anche in sistemi omogenei come sistemi HPC e infrastrutture di cloud computing.

L'architettura di D-MASON è stata presentata nelle seguenti pubblicazioni:

- Cordasco G., Spagnuolo C. and Scarano V. *Toward the new version of D-MASON: Efficiency, Effectiveness and Correctness in Parallel and Distributed Agent-based Simulations.* 1st IEEE Workshop on Parallel and Distributed Processing for Computational Social Systems. IEEE International Parallel & Distributed Processing Symposium 2016.
- Cordasco G., De Chiara R., Mancuso A., Mazzeo D., Scarano V. and Spagnuolo C. *Bringing together efficiency and effectiveness in distributed simulations: the experience with D-MASON.* SIMULATION: Transactions of The Society for Modeling and Simulation International, June 11, 2013.
- Cordasco G., De Chiara R., Mancuso A., Mazzeo D., Scarano V. and Spagnuolo C. *A Framework for distributing Agent-based simulations.* Ninth International Workshop Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms of Euro-Par 2011 conference.

Uno degli strati architetturali di D-MASON che ne determina le prestazioni, è il il Communication Layer, il quale offre le funzionalità di comunicazione tra tutte le entità coinvolte nel calcolo. La comunicazione in D-MASON è basata sul paradigma Publish/Subscribe (PS). Al fine di soddisfare la flessibilità e la scalabilità richiesta, vengono fornite due strategie di comunicazione, una centralizzata (utilizzando Java Message Service) e una decentralizzata (utilizzando Message Passing Interface). La comunicazione in sistemi omogenei è sempre basata su PS ma utilizza lo standard Message Passing Interface (MPI). Al fine di utilizzare MPI in Java, lo strato di comunicazione di D-MASON è implementato sfruttando un binding Java a MPI. Tale soluzione non permette però l'utilizzo di tutte le funzionalità di MPI. Al tal proposito molteplici soluzioni sono stare progettate e implementate, e sono presentate nelle seguenti pubblicazioni:

- Cordasco G., Milone F., Spagnuolo C. and Vicidomini L. *Exploiting D-MASON on Parallel Platforms: A Novel Communication Strategy* 2st Workshop on Parallel and Distributed Agent-Based Simulations of Euro-Par 2014 conference.
- Cordasco G., Mancuso A., Milone F. and Spagnuolo C. *Communication strategies in Distributed Agent-Based Simulations: the experience with D-MASON* 1st Workshop on Parallel and Distributed Agent-Based Simulations of Euro-Par 2013 conference.

D-MASON offre anche meccanismi per la visualizzazione centralizzata e la raccolta di informazioni in simulazioni distribuite (tramite il Visualization Layer). I risultati ottenuti sono stati presentati nella seguente pubblicazione:

- Cordasco G., De Chiara R., Raia F., Scarano V., Spagnuolo C. and Vicidomini L. *Designing Computational Steering Facilities for Distributed Agent Based Simulations.* Proceedings of the ACM SIGSIM Conference on Principles of Advanced Discrete Simulation 2013.

Quando si parla di simulazioni distribuite una delle principali problematiche è il bilanciamento del carico. D-MASON offre, nel Distributed Simulation Layer, meccanismi per il partizionamento dinamico e il bilanciamento del carico. D-MASON utilizza la tecnica del field partitioning per suddividere il lavoro tra le entità del sistema distribuito. La tecnica di field partitioning consente di ottenere un buon equilibrio tra il bilanciamento del carico e l'overhead di comunicazione.

Molti modelli di simulazione non sono basati su spazi 2/3-dimensionali ma bensì modellano le relazioni tra gli agenti utilizzando strutture dati grafo. In questi casi la tecnica di field partitioning non garantisce soluzioni che consentono di ottenere buone prestazioni. Per risolvere tale problema, D-MASON fornisce particolari soluzioni per simulazioni che utilizzano i grafi per modellare le relazioni tra gli agenti.

I risultati conseguiti sono stati presentati nella seguente pubblicazione:

- Antelmi A., Cordasco G., Spagnuolo C. and Vicidomini L.. *On Evaluating Graph Partitioning Algorithms for Distributed Agent Based Models on Networks*. 3rd Workshop on Parallel and Distributed Agent-Based Simulations of Euro-Par 2015 conference.

Il metodo di field partitioning consente il partizionamento di campi Euclidei mono e bi-dimensionali; tale approccio è anche conosciuto con il nome di partizionamento uniforme. In alcuni casi, come ad esempio simulazioni che utilizzano Geographical Information System (GIS), il metodo di partizionamento uniforme non è in grado di garantire buone prestazioni, a causa del posizionamento non bilanciato degli agenti sul campo di simulazione. In questi casi, D-MASON offre un meccanismo di partizionamento non uniforme (inspirato alla struttura dati Quad-Tree), presentato nelle seguenti pubblicazioni:

- Lettieri N., Spagnuolo C. and Vicidomini L.. *Distributed Agent-based Simulation and GIS: An Experiment With the dynamics of Social Norms*. 3rd Workshop on Parallel and Distributed Agent-Based Simulations of Euro-Par 2015 conference.
- G. Cordasco and C. Spagnuolo and V. Scarano. *Work Partitioning on Parallel and Distributed Agent-Based Simulation*. IEEE Workshop on

Parallel and Distributed Processing for Computational Social Systems of International Parallel & Distributed Processing Symposium, 2017.

Inoltre, D-MASON è stato esteso allo scopo di fornire un'infrastruttura Simulation-as-a-Service (SIMaaS), che semplifica il processo di esecuzione di simulazioni distribuite in un ambiente di Cloud Computing. D-MASON nella sua versione più recente offre uno strato software di management basato su web, che ne consente estrema facilità d'uso in ambienti Cloud. Utilizzando il System Management, D-MASON è stato sperimentato sull'infrastruttura Cloud Amazon EC2 confrontando le prestazioni in questo ambiente cloud con un sistema HPC. I risultati ottenuti sono stati presentati nella seguente pubblicazione:

- M Carillo, G Cordasco, F Serrapica, C Spagnuolo, P. Szufel, and L. Vicidomini. *D-Mason on the Cloud: an Experience with Amazon Web Services*. 4rd Workshop on Parallel and Distributed Agent-Based Simulations of Euro-Par 2016 conference.

Linguaggi Paralleli. La proposta di un'architettura, la quale consente di invocare il codice per Java Virtual Machine (JVM) da codice scritto in linguaggio C. Swift/T è un linguaggio di scripting parallelo per sviluppare applicazioni altamente scalabili in ambienti paralleli e distribuiti. Swift/T è l'implementazione del linguaggio Swift per ambienti HPC. Swift/T migliora il linguaggio Swift, consentendo la scalabilità fino a 500 task per secondo, il bilanciamento del carico, strutture dati distribuite, e dataflow task execution.

Swift/T consente di invocare nativamente codice scritto in altri linguaggi (come Python, R, Julia e C) utilizzando particolari funzioni definite come leaf function. Il trend attuale di molti produttori di sistemi di supercalcolo (come Cray Inc.), è quello di offrire processori che supportano JVM. Considerato ciò in questa tesi viene presentato il metodo adottato in Swift/T per l'invocazione di linguaggi per JVM (come Java, Clojure, Scala, Groovy, JavaScript) da Swift/T. A tale scopo è stato realizzato un binding C per l'invocazione e la gestione di codice per JVM. Questa soluzione è stata utilizzata in Swift/T (dalla versione 1.0) per estendere il supporto del linguaggio anche a linguaggi per JVM. Il codice sviluppato è stato rilasciato sotto licenza open source ed è disponibile in un repository pubblico su GitHub.

Frameworks. La proposta di due tool che sfruttano la potenza di calcolo di sistemi distribuiti per migliorare l'efficacia e l'efficienza di strategie di Simulation Optimization. Simulation Optimization (SO) si riferisce alle tecniche utilizzate per l'individuazione dei parametri di un modello complesso che minimizzano (o massimizzano) determinati criteri, i quali possono essere computati solo tramite l'esecuzione di una simulazione. A causa dell'elevata dimensionalità dello spazio dei parametri, della loro eterogeneità e, della natura stocastica della funzione di

valutazione, la configurazione di tali sistemi è estremamente onerosa dal punto di vista computazionale. In questo lavoro sono presentati due framework per SO.

Il primo framework è *SOF: Zero Configuration Simulation Optimization Framework on the Cloud*, progettato per l'esecuzione del processo SO in ambienti di cloud computing. SOF è basato su Apache Hadoop ed è presentato nella seguente pubblicazione:

- Carillo M., Cordasco G., Scarano V., Serrapica F., Spagnuolo C. and Szufel P. *SOF: Zero Configuration Simulation Optimization Framework on the Cloud.* Parallel, Distributed, and Network-Based Processing 2016.

Il secondo framework è *EMEWS: Extreme-scale Model Exploration with Swift/T*, progettato per eseguire processi SO in sistemi HPC. Entrambi i framework sono stati sviluppati principalmente per ABS. In particolare EMEWS è stato sperimentato utilizzando il toolkit ABS chiamato Repast. Nella sua prima versione EMEWS non supportava simulazioni scritte in MASON e NetLogo. In questo lavoro di tesi sono descritte alcune funzionalità di EMEWS che consentono il supporto a tali simulazioni. EMEWS e alcuni casi d'uso sono presentati nella seguente pubblicazione:

- J. Ozik, N. T. Collier, J. M. Wozniak and C. Spagnuolo *From Desktop To Large-scale Model Exploration with Swift/T.* Winter Simulation Conference 2016.

Architetture. La proposta di un'architettura open source per la visualizzazione web di dati dinamici. Tale architettura si basa sul paradigma di Edge-centric Computing; la visualizzazione dei dati è eseguita lato client, garantendo in questo modo l'affidabilità dei dati, la privacy e la scalabilità in termini di numero di visualizzazioni concorrenti. L'architettura è stata utilizzata all'interno della piattaforma sociale SPOD (Social Platform for Open Data), ed è stata presentata nelle seguenti pubblicazioni:

- G. Cordasco, D. Malandrino, P. Palmieri, A. Petta, D. Pirozzi, V. Scarano, L. Serra, C. Spagnuolo, L. Vicidomini *A Scalable Data Web Visualization Architecture.* Parallel, Distributed, and Network-Based Processing 2017.
- G. Cordasco, D. Malandrino, P. Palmieri, A. Petta, D. Pirozzi, V. Scarano, L. Serra, C. Spagnuolo, L. Vicidomini *An Architecture for Social Sharing and Collaboration around Open Data Visualisation.* In Poster Proc. of the 19th ACM conference on "Computer-Supported Cooperative Work and Social Computing 2016.
- G. Cordasco, D. Malandrino, P. Palmieri, A. Petta, D. Pirozzi, V. Scarano, L. Serra, C. Spagnuolo, L. Vicidomini *An extensible architecture for an ecosystem of visualization web-components for Open Data* Maximising interoperability Workshop— core vocabularies, location-aware data and more 2015.