UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Corso di Laurea Magistrale in Ingegneria Informatica

Implementazione e valutazione di uno scheduler per l'esecuzione di workflow data-intensive

Relatori:
Prof. ***Domenico Talia***
Ing. Fabrizio Marozzo

Candidato:
Giovambattista Moraca
Matr. 195289

Anno Accademico 2019/2020

Indice

Introduzione 1

1 Stato dell'arte 4

1.1 Kosar 2008: lo scheduler Stork 5

1.1.1 Metadati 6

1.1.2 Scheduler 7

1.2 Wang 2014: il sistema MATRIX 10

1.2.1 Metadati 11

1.2.2 Scheduler 12

1.3 Wei 2005: uno scheduler plugin per LSF 19

1.3.1 Metadati 20

1.3.2 Scheduler 22

1.4 Acevedo 2017: scheduling tramite un algoritmo di critical path 26

1.4.1 metadati 27

1.4.2 Scheduler 32

1.5 Tabella Comparativa 35

2 Simulatore 36

2.1 CloudSim 36

2.1.1 Architettura a livelli 37

2.1.2 Componenti principali 38

2.1.3 Core Simulation Framework 41

2.1.4 Processamento interno della simulazione 43
2.2 WorkflowSim 46
2.2.1 Componenti 47
2.2.2 Scheduling 49
2.2.3 Overhead e fallimenti 51

3 Implementazione 54
3.1 Planner 57
3.1.1 Critical path method 57
3.1.2 CPPlanningAlgorithm 62
3.2 Scheduling 65
3.2.1 WorkStealer 68
3.2.2 MatrixSchedulingAlgorithm 69
3.3 Valutazioni 73

Conclusioni 81

Bibliografia 82

Introduzione

Negli ultimi anni, gli esperimenti e le applicazioni scientifiche moderne hanno fatto un uso sempre più intenso dei dati. Esperimenti quali simulazioni fisiche ad alte energie, mappatura genomica e modellazione climatica, generano un enorme volume di dati che raggiunge le centinaia di terabyte.

Al fine di elaborare questi dati, gli scienziati si stanno orientando verso risorse distribuite di terze parti, che forniscono loro la potenza di elaborazione e la capacità di archiviazione necessaria per portare avanti la ricerca.

Ma l'uso di questo tipo di risorse impone nuove sfide, specialmente con l'arrivo imminente di macchine exascale [1]. Al giorno d'oggi, infatti, i più potenti supercomputer al mondo sono in grado di eseguire un numero nell'ordine dei quadrilioni (10^{15}) di calcoli al secondo (petascale), un sistema exascale raggiunge gli exaFLOPS (10^{18} floating point operations per second), ed è quindi in grado di analizzare rapidamente enormi volumi di dati ed eseguire simulazioni e computazioni in tempi di gran lunga più brevi.

Il rovescio della medaglia sta nel fatto che anche se molto più efficienti, risultano anche molto più complicati da utilizzare, un numero così elevato di core (oltre le centinaia di migliaia) e l'utilizzo di una tale mole di dati, rende inadatte le tecniche di programmazione attuali.

Il costo energetico relativo allo spostamento dei dati, sta diventando un fattore dominante, poiché il costo energetico legato alla computazione,

sta diminuendo a un tasso più rapido rispetto a quello di spostare dati su chip. Si è arrivati al punto in cui i miglioramenti prestazionali in termini di efficienza computazionale, vengono messi in ombra dai costi energetici necessari a portare i dati ai core.

Gli ambienti di programmazione odierni, si basano sull'idea che la computazione, sia la componente più costosa, ma come abbiamo visto, ci stiamo rapidamente spostando verso una situazione in cui la computazione risulta economica e disponibile ovunque (ubiquitous computing [2]) mentre lo spostamento dei dati risulta prevalere sui costi energetici. Questi sviluppi hanno ribaltato i presupposti che stanno alla base della programmazione e fanno presagire un passaggio da un paradigma compute-centric, incentrato sulla computazione, per la programmazione di sistemi informatici ad uno data-centric [3], incentrato sui dati.

Gli attuali modelli di programmazione compute-centric si basano fondamentalmente su un'astrazione di macchina, in cui gli elementi di elaborazione all'interno di un nodo sono equidistanti. I modelli data-centric, per contro, utilizzano un'astrazione che descrive come i dati sono disposti sul sistema, ed eseguono la computazione sui dati dove essi risiedono. Ecco perché sono necessarie astrazioni adeguate per sistemi che fanno un uso molto intenso della concorrenza e tecniche di data affinity, che minimizzano il movimento dei dati distribuendo la computazione dove i dati vengono generati, per utilizzare al meglio i sistemi futuri.

Nel mio lavoro di tesi mi concentro su uno dei problemi principali della data affinity: lo scheduling. Gli scheduler che tengono in considerazione la posizione dei dati in memoria e la utilizzano per assegnare in maniera adeguata i processi da eseguire ai nodi computazionali, sono definiti data-aware ed è proprio su questi ultimi che si focalizza il mio elaborato.

La tesi è divisa in tre capitoli:

- Nel primo vengono presi in esame gli scheduler allo stato dell'arte, prestando particolare attenzione alle loro metodologie di data affinity;

- Il secondo capitolo mostra il funzionamento del simulatore WorkflowSim [4] che ho utilizzato per realizzare e simulare una mia implementazione di scheduler data-aware;

- L'ultimo capitolo contiene una descrizione dell'implementazione dello scheduler realizzata utilizzando tecniche presenti negli scheduler analizzati nel primo capitolo, insieme con i risultati dei test effettuati mediante simulazioni su WorkflowSim, nelle quali lo scheduler è stato confrontato con gli altri presenti nel simulatore.

Bibliografia

[1] G. Da Costa, T. Fahringer, J. A. R. Gallego, I. Grasso, A. Hristov, H. D. Karatza, A. Lastovetsky, F. Marozzo, D. Petcu, G. L. Stavrinides, *et al.*, "Exascale machines require new programming paradigms and runtimes," *Supercomputing frontiers and innovations*, vol. 2, no. 2, pp. 6–27, 2015.

[2] "Ubiquitous computing." https://it.wikipedia.org/wiki/Ubiquitous_computing.

[3] D. Talia, P. Trunfio, F. Marozzo, L. Belcastro, J. Garcia-Blas, D. del Rio, P. Couvée, G. Goret, L. Vincent, A. Fernández-Pena, *et al.*, "A novel data-centric programming model for large-scale parallel systems,"

[4] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *2012 IEEE 8th International Conference on E-Science*, pp. 1–8, IEEE, 2012.

[5] T. Kosar and M. Balman, "A new paradigm: Data-aware scheduling in grid computing," *Future Generation Computer Systems*, vol. 25, no. 4, pp. 406–413, 2009.

[6] M. Solomon, "The classad language reference manual version 2.4," 2004.

[7] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed

systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[8] "HTCondor DAGMan." https://research.cs.wisc.edu/htcondor/dagman/dagman.html.

[9] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-g: A computation management agent for multi-institutional grids," *Cluster Computing*, vol. 5, no. 3, pp. 237–246, 2002.

[10] K. Wang, X. Zhou, T. Li, D. Zhao, M. Lang, and I. Raicu, "Optimizing load balancing and data-locality with data-aware scheduling," in *2014 IEEE International Conference on Big Data (Big Data)*, pp. 119–128, IEEE, 2014.

[11] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, "Zht: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pp. 775–787, IEEE, 2013.

[12] "Platform LSF." https://en.wikipedia.org/wiki/Platform_LSF.

[13] J. Cao and T. Washio, "Optimization of docking conformations using grid datafarm," 2003.

[14] X. Wei, W. W. Li, O. Tatebe, G. Xu, L. Hu, and J. Ju, "Implementing data aware scheduling in gfarm (r) using lsf (tm) scheduler plugin mechanism.," *GCA*, vol. 5, pp. 3–5, 2005.

[15] C. Acevedo, P. Hernández, A. Espinosa, and V. Méndez, "A critical path file location (cpfl) algorithm for data-aware multiworkflow scheduling on hpc clusters," *Future Generation Computer Systems*, vol. 74, pp. 51–62, 2017.

[16] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[17] "Critical Path Image." https://media.geeksforgeeks.org/wp-content/uploads/20190510203214/aon4.jpg.

[18] "Heterogeneous Earliest Finish Time." https://en.wikipedia.org/wiki/Heterogeneous_Earliest_Finish_Time.

[19] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *2008 third workshop on workflows in support of large-scale science*, pp. 1–10, IEEE, 2008.