Striking while the Iron is Hot:

Using the Predictive Skeletal Information to Timely Improve Parallel Performance in Grids

Horacio González–Vélez

h.gv@ed.ac.uk

Introduction

Algorithmic skeletons abstract commonly used patterns of parallel computation, communication, and interaction [1]. They present a top-down transformational approach where programs are formed from skeleton nesting. By demonstrating a predictable communication and computation structure, they provide a preponderant foundation for performance modelling and estimation.

Computational grids pose a challenge to known distributed systems techniques as a result of their dynamism. One of the most prominent research areas concerns the availability of proved programming paradigms with special emphasis on the performance side. Thus, adaptable performance improvement techniques have been the subject of intense scrutiny.

Motivation

Compile-time optimisers formulate static decisions about the expected behaviour of an application. On the other hand, run-time optimisers do not generally possess direct knowledge of the structure of the application. They lack specific information on its data and control flows and their operation is normally driven by load-balancing criteria.

Although different parallel solutions have traditionally exhibited skeleton-like constructs such as task farming, optimisers do not necessarily employ the forecasting information of the constructs but rather modify the scheduler, or do not decouple entirely the application structure from its behaviour.

Skeletons provide a clear and consistent meaning across platforms while their associated behaviour depends on the particular implementation. They possess a crucial property which favours performance optimisation: Their **structured and predictable behaviour for a given program**. Nevertheless, scant research has been conducted on improving the skeletal performance by actively using this information from a systems infrastructure perspective.

Aims

We plan to garner a set of adaptive optimisation techniques to boost the skeletal performance in the Grid. This set will address pragmatic issues in synchronisation, communication and processing for skeletons in a grid. Our approach is:

Timely Adjusts to the grid conditions at execution time

Integrated Incorporates skeletal characteristics at compile time

Parametric Provides criteria to differentiate conditions in the Grid

Method

A Skeletal Processor Farm, programmed in C with PACX-MPI inter-node communications, has been initially used to compare the effectiveness of the proposed approach. **This PF adapts the task size using the knowledge on the nodes and the network.** It employs not only the auto-scheduling features of the master-slave pattern, but also the Network Weather System (NWS) [3] monitoring capabilities.

In practical terms, the PF transforms an input data set into an output set by applying a given function to each element of the input data set. In order to do this, the farmer distributes a certain number of elements (task size) to each worker. Thus, the task size defines the granularity of the farm, and hence the scheduling of the PF. The results are then collected into the output data set.

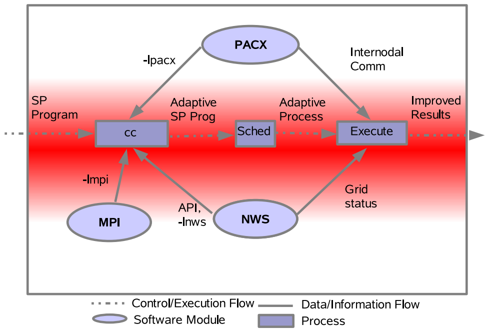
The adaptive strategy, central part of this work, defines the task size by taking into account four factors:

Current CPU CPU fraction usable by a running process

Available CPU CPU fraction allocatable to a new process

Latency Time to send a TCP message from the farmer to a worker

Bandwidth Speed which data can be transmitted to/from the farmer and a worker



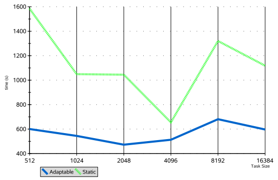
Overview of the Adaptable Skeletal Processor Farm

The main intention of this initial experiment is to help us to discretise and bound the parameters involved, reducing the number of combinations. Although measurements are supplied at execution time, the use of skeletons allows us to make assumptions on the input and output sets that permit a more effective allocation of computation and communication resources.

Results

The initial deployment utilises a sandbox grid environment composed by 32 processors distributed into two 16-node Beowulf clusters (`bw240.inf` and `bw530.inf`). Twenty seven workers, on the same number of non-dedicated processors, were allocated to compute, and the remaining five processors were used as the farmer and a two-pair communication assembly.

Each worker performed $O(10^9)$ daxpy operations per task element. The timings are compared against a typical, static processor farm. All nodes were on non-dedicated mode during all the experiments, and their interconnection channels did not have any bandwidth reservation.



Comparison of Execution Times: Adaptive vs. Static PF

It is important to notice that **the adaptive model surpasses the static one in every single task size**. The variations in the execution times in both models are defined by the non-deterministic nature of this grid environment.

Future Directions

- Develop a more accurate adaptiveness strategy through a more comprehensive experimentation. A cell biology code is being tinkered with [2]
- Deploy a faster distribution of tasks by incorporating buffering
- Improve the model by incorporating new indicators such as task termination time and CPU capacity

References

[1] Cole, M. Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming. *Parallel Comput. 30*, 3 (2004), 389–406.

[2] Gonzalez-Velez, V., and Gonzalez-Velez, H. A grid-based stochastic simulation of unitary and membrane Ca^{2+} currents in spherical cells. In *The 18th IEEE Int Symp on Computer-Based Medical Systems* (Dublin, Ireland, June 2005), IEEE CS. Accepted for publication.

[3] Wolski, R., Spring, N., and Hayes, J. The Network Weather Service: A distributed resource performance forecasting service for metacomputing. *Future Gener. Comput. Syst. 15*, 5–6 (1999), 757–768.

School of informatics