Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Tom CORNEBIZE

Thèse dirigée par **Arnaud LEGRAND**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
et de l'École Doctorale **MSTII**

Le Titre de la Thèse

The English Title

Thèse soutenue publiquement le **1er janvier 1970**,
devant le jury composé de :

I dedicate this thesis to my grumpy cat.

> *Elle est où la poulette ?*
>
> — **Kadoc De Vannes**

Remerciements

(Acknowledgments)

I would like to thank everyone, except from Dobby the free elf.

Merci public !

Abstract / Résumé

Abstract

The English abstract.

Résumé

Le résumé en français.

Contents

Introduction

To introduce my work, I will write a nice introduction in the following. Citation example for the Top500 website [@top500] and some random paper [Gra69].

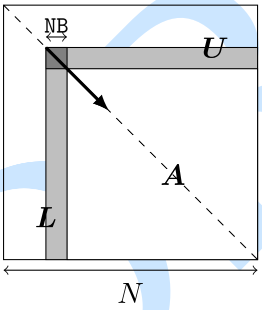
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2 Performance prediction through simulation: the HPL case

The work presented in this chapter has been published at a conference[CLH19] and has been submitted for publication in a journal. The content of this chapter is therefore a near-verbatim copy of these articles.

2.1 High Performance Linpack

2.1.1 The benchmark



```
allocate and initialize $A$;
for $k = N$ to $0$ step $NB$ do
    allocate the panel;
    factor the panel;
    broadcast the panel;
    update the sub-matrix;
end
```

Figure 2.1: Overview of High Performance Linpack

In this work, we use the freely-available reference-implementation of HPL[Pet+16], which relies on MPI. HPL implements a matrix factorization based on a right-looking variant of the LU factorization with row partial pivoting and allows multiple look-ahead depths. The principle of the factorization is depicted in Figure2.1. It consists of a series of panel factorizations followed by an update of the trailing sub-matrix. HPL uses a two-dimensional block-cyclic data distribution of A and implements several custom MPI collective communication algorithms to efficiently overlap communications with computations. The main parameters of HPL are:

- N is the order of the square matrix A.

- NB is the *blocking factor*, i.e. the granularity at which HPL operates when panels are distributed or worked on.

- P and Q denote the number of process rows and the number of process columns, respectively.

- RFACT determines the panel factorization algorithm. Possible values are Crout, left- or right-looking.

- SWAP specifies the swapping algorithm used while pivoting. Two algorithms are available: one based on *binary exchange* (along a virtual tree topology) and the other one based on a *spread-and-roll* (with a higher number of parallel communications). HPL also provides a panel-size threshold triggering a switch from one variant to the other.

- BCAST sets the algorithm used to broadcast a panel of columns over the process columns. Legacy versions of the MPI standard only supported non-blocking point-to-point communications, which is why HPL ships with in total 6 self-implemented variants to overlap the time spent waiting for an incoming panel with updates to the trailing matrix: ring, ring-modified, 2-ring, 2-ring-modified, long, and long-modified. The modified versions guarantee that the process right after the root (i.e. the process that will become the root in the next iteration) receives data first and does not further participate in the broadcast. This process can thereby start working on the panel as soon as possible. The ring and 2-ring versions each broadcast along the corresponding virtual topologies while the long version is a *spread and roll* algorithm where messages are chopped into Q pieces. This generally leads to better bandwidth exploitation. The ring and 2-ring variants rely on MPI_Iprobe, meaning they return control if no message has been fully received yet, hence facilitating partial overlap of communication with computations. In HPL 2.1 and 2.2, this capability has been deactivated for the long and long-modified algorithms. A comment in the source code states that some machines apparently get stuck when there are too many ongoing messages.

- DEPTH controls how many iterations of the outer loop can overlap with each other.

The sequential complexity of this factorization is

$$\text{flop}(N) = \frac{2}{3}N^3 + 2N^2 + \text{‰}(N)$$

where N is the order of the matrix to factorize. The time complexity can be approximated by

$$T(N) \approx \frac{\left(\frac{2}{3}N^3 + 2N^2\right)}{P \cdot Q \cdot w} + \Theta((P+Q) \cdot N^2)$$

where w is the flop rate of a single node and the second term corresponds to the communication overhead which is influenced by the network capacity and the previously listed parameters (`RFACT`, `SWAP`, `BCAST`, `DEPTH`, ...) and is very difficult to predict.

2.1.2 Typical runs on a supercomputer

Although the TOP500 reports precise information about the core count, the peak performance and the effective performance, it provides almost no information on how (software versions, HPL parameters, etc.) this performance was achieved. Some colleagues agreed to provide us with the HPL configuration they used and the output they submitted for ranking (see Table2.1). In June 2013, the Stampede supercomputer at TACC was ranked 6th in the TOP500 by achieving $5168.1\,\mathrm{TFlop\,s^{-1}}$. In November 2017, the Theta supercomputer at ANL was ranked 18th with a performance of $5884.6\,\mathrm{TFlop\,s^{-1}}$ but required a 28-hour run on the whole machine. Finally, we ran HPL ourselves on a Grid'5000 cluster named Dahu whose software stack could be fully controlled.

Table 2.1: Typical runs of HPL

	Stampede@TACC	Theta@ANL	Dahu@G5K
`Rpeak`	$8520.1\,\mathrm{TFlop\,s^{-1}}$	$9627.2\,\mathrm{TFlop\,s^{-1}}$	$62.26\,\mathrm{TFlop\,s^{-1}}$
N	3,875,000	8,360,352	500,000
`NB`	1024	336	128
$P \times Q$	77×78	32×101	32×32
`RFACT`	Crout	Left	Right
`SWAP`	Binary-exch.	Binary-exch.	Binary-exch.
`BCAST`	Long modified	2 Ring modified	2 Ring
`DEPTH`	0	0	1
`Rmax`	$5168.1\,\mathrm{TFlop\,s^{-1}}$	$5884.6\,\mathrm{TFlop\,s^{-1}}$	$24.55\,\mathrm{TFlop\,s^{-1}}$
Duration	2 hours	28 hours	1 hour
Memory	120 TB	559 TB	2 TB
MPI ranks	1/node	1/node	1/core

The performance typically achieved by supercomputers (`Rmax`) needs to be compared to the much larger peak performance (`Rpeak`). This difference can be attributed to the node usage, to the MPI library, to the network topology that may be unable to deal with the intense communication workload, to load imbalance among nodes (e.g. due to a defect, system noise, ...), to the algorithmic structure of HPL, etc. All these factors make it difficult to know precisely what performance to expect without running the application at scale. It is clear that due to the level of complexity of both HPL and the underlying hardware, simple performance models (analytic expressions based on N, P, Q and estimations of platform characteristics) may be able to provide trends but can by no means accurately predict the performance for each configuration (e.g. consider the exact effect of HPL's six different broadcast algorithms on network contention). Additionally, these expressions do not allow engineers to improve the performance through actively identifying performance bottlenecks. For complex optimizations such as partially non-blocking collective communication algorithms intertwined with computations, a very faithful modeling of both the application and the platform is required. One goal of this thesis was to simulate systems at the scale of Stampede. Given the scale of this scenario (3,785 steps on 6,006 nodes in two hours), detailed simulations quickly become intractable without significant effort.

2.2 Simgrid/SMPI and the other simulators

some text...

2.3 Emulating HPL at large scale

some text...

2.4 Modeling HPL kernels and communications

some text...

2.5 Validation

some text...

2.6 Sensibility analysis

some text...

3 Experimental control

3.1 Experimental Testbed and Experiment Engines

some text...

3.2 Randomizing matters!

some text...

3.3 Performance non-regression tests

some text...

4 Conclusion

Your beautiful conclusion. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Bibliography

[@top500] *TOP500 Website*. URL: `https://www.top500.org/` (visited on Sept. 7, 2020). cit. on p. 1

[CLH19] Tom Cornebize, Arnaud Legrand, and Franz C Heinrich. “Fast and Faithful Performance Prediction of MPI Applications: the HPL Case Study”. In: *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. 2019 IEEE International Conference on Cluster Computing (CLUSTER). Albuquerque, United States, Sept. 2019. cit. on p. 3

[Gra69] Ronald L. Graham. “Bounds on multiprocessing timing anomalies”. In: *SIAM journal on Applied Mathematics* 17.2 (1969), pp. 416–429. cit. on p. 1

[Pet+16] Antoine Petitet, Clint Whaley, Jack Dongarra, Andy Cleary, and Piotr Luszczek. *HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*. `http://www.netlib.org/benchmark/hpl`. Version 2.2. Feb. 2016. cit. on p. 3

List of Figures

List of Tables

Abstract

The English abstract.

Résumé

Le résumé en français.