Разработка параллельных алгоритмов для современных вычислительных систем – матанализ вместо арифметики

А.Л. Ластовецкий

University College Dublin, Ирландия

Летняя Суперкомпьютерная Академия
1 июля 2016

Outline

Mainstream Performance Optimization Algorithms

Outline

Mainstream Performance Optimization Algorithms

Application performance profiles on modern platforms

Outline

Mainstream Performance Optimization Algorithms

Application performance profiles on modern platforms

Challenges and Benefits of Accepting Reality

- Experimental Results

Mainstream approach to performance modelling

Most of algorithms for performance optimization are based on very simple models:

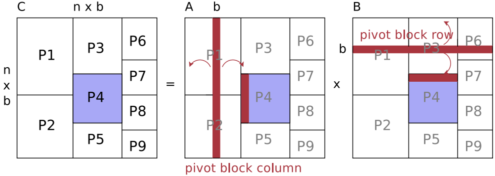
- Scheduling algorithms
- Load balancing algorithms
- Data partitioning algorithms
- Task mapping algorithms

They assume the speed of processing element to be constant.

Matrix partitioning

Matrix partitioning problem for parallel matrix multiplication on heterogeneous platforms*

- Input: constant processor speeds
- Matrices partitioned so that
  - Area of the rectangle proportional to the speed
  - Volume of communication minimized



Maths used by algorithms solving this problem do not go beyond basic arithmetics.

* Beaumont, O. et al: Matrix Multiplication on Heterogeneous Platforms. IEEE Trans. Parallel Distrib. Syst. 2001

Traditional Dynamic Load Balancing Algorithms

A routine has n computational units distributed across p processors.
Processor P_i has d_i units such that $n = \sum_{i=1}^{p} d_i$
Initially $d_i^0 = n/p$
At each iteration

1. Execution times for this iteration measured and gathered to root
2. if relative difference between times $\leq \epsilon$
   then no balancing needed
   else new distribution is calculated as:
   $d_i^{k+1} = n \times s_i^k / \sum_{j=1}^{p} s_j^k$ where speed $s_i^k = d_i^k / t_i(d_i^k)$
3. new distributions d_i^{k+1} broadcast to all processors and where necessary data is redistributed accordingly.

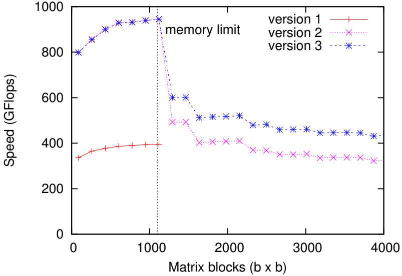
Domain decomposition in CFD

Parallel CFD packages such as OpenFOAM use graph/mesh partitioning libraries for domain decomposition

- MeTiS, Scotch, etc.
- Input - vector of positive constants representing the relative volume of computation to be performed by each processor

The Reality: Matrix Multiplication

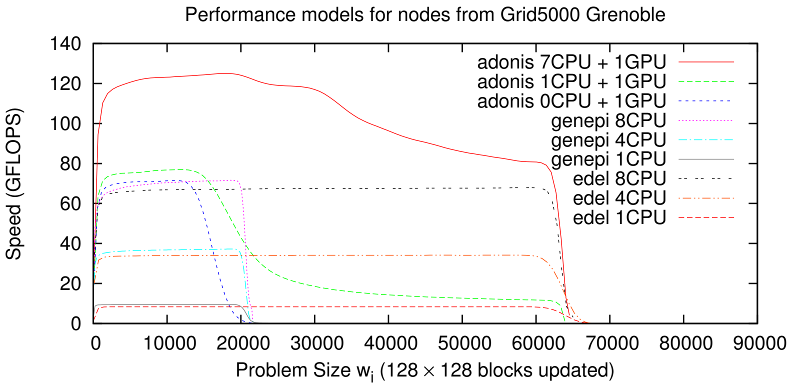
Functional Performance Models of GPU



- g(x) (version 1): naive kernel
- g(x) (version 2): accumulate intermediate result + out-of-core
- g(x) (version 3): version 2 + overlap data transfers and kernel executions

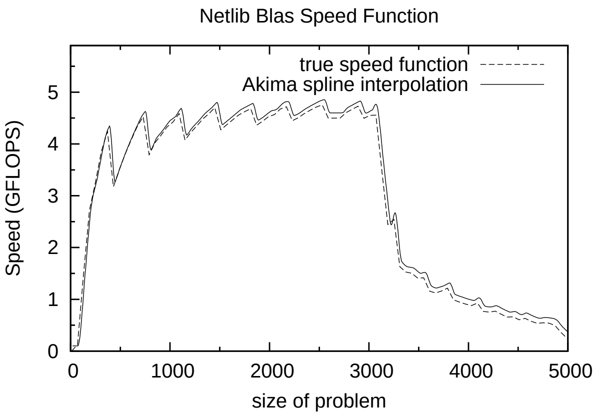
The Reality: Matrix Multiplication

Experimental results for Grid'5000 nodes



The Reality: Matrix Multiplication

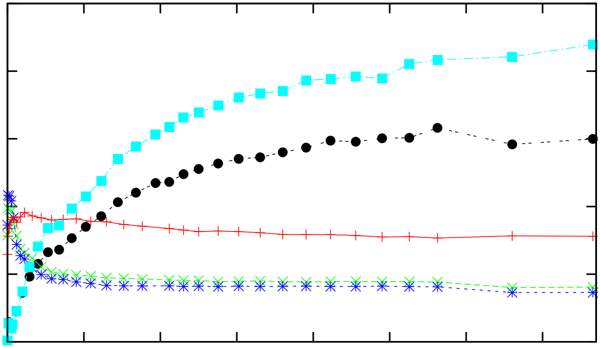
Experimental results for Netlib BLAS dgemm*



* Rychkov, V. et al: Using Multidimensional Solvers for Optimal Data Partitioning on Dedicated Heterogeneous HPC Platforms. PaCT'2011

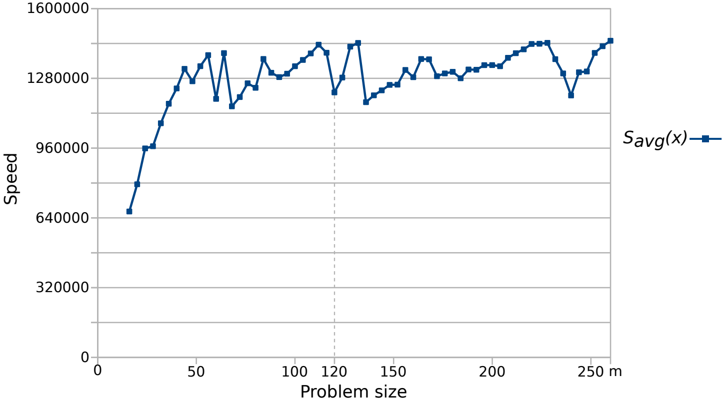
The Reality: Computational Fluid Dynamics

Speed functions of the CG solver built in different configurations on an Adonis node (GFlops against the number of control volumes)



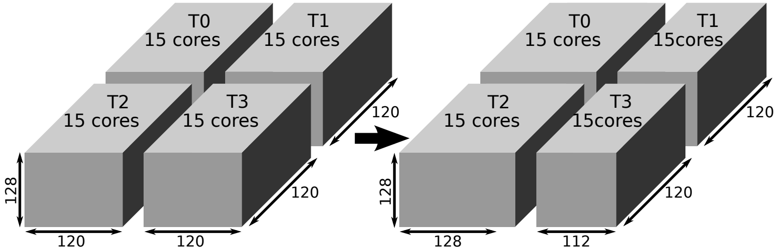
The Reality: Computational Fluid Dynamics

Experimental results for MPDATA on Intel Xeon Phi
(domain size $120 \times m \times 128$)



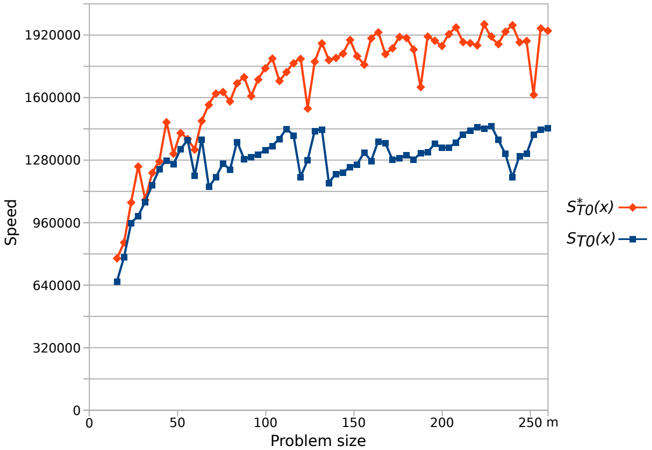
The Reality: Computational Fluid Dynamics

Experimental results for MPDATA on Intel Xeon Phi



The Reality: Computational Fluid Dynamics

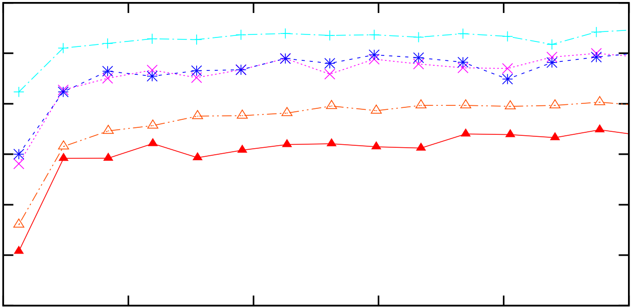
More experimental results for MPDATA on Intel Xeon Phi: the impact of resource sharing



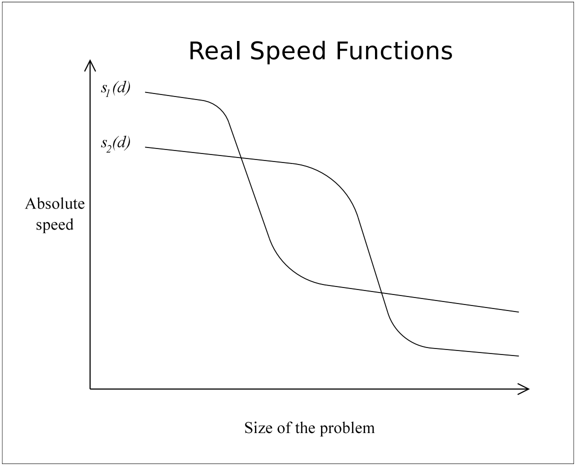
The Reality: Matrix Multiplication

The impact of resource sharing: the speed of a CPU core built in different configurations

- $s_1(x)$, $s_6(x)$, $s_{12}(x)$, $S_6(6x)/6$, $S_{12}(12x)/12$

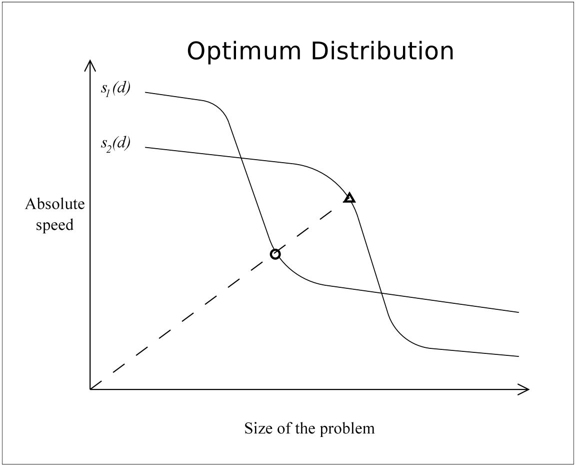


Straightforward dynamic load balancing may not balance



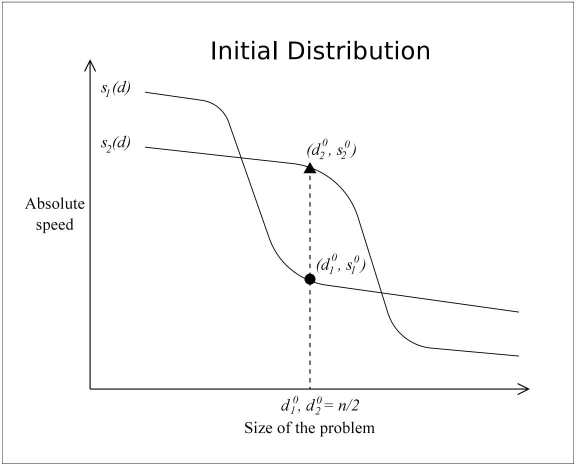
* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Straightforward dynamic load balancing may not balance



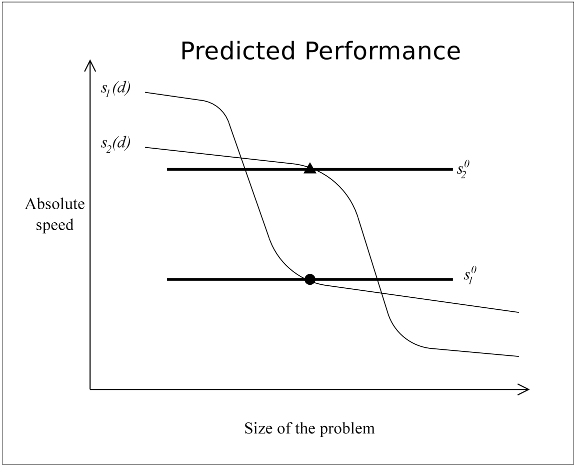
* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Straightforward dynamic load balancing may not balance



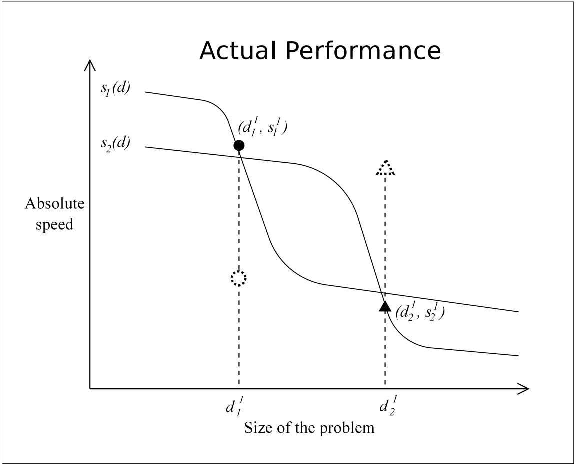
* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Straightforward dynamic load balancing may not balance



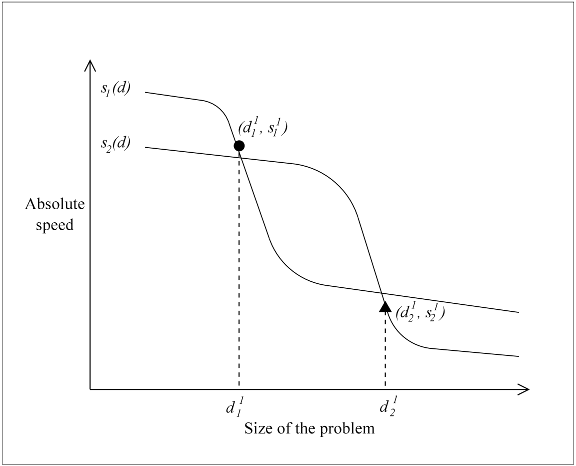
* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Straightforward dynamic load balancing may not balance



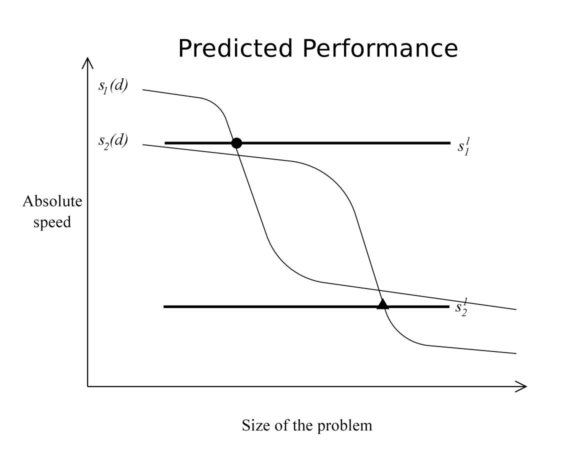
* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Straightforward dynamic load balancing may not balance



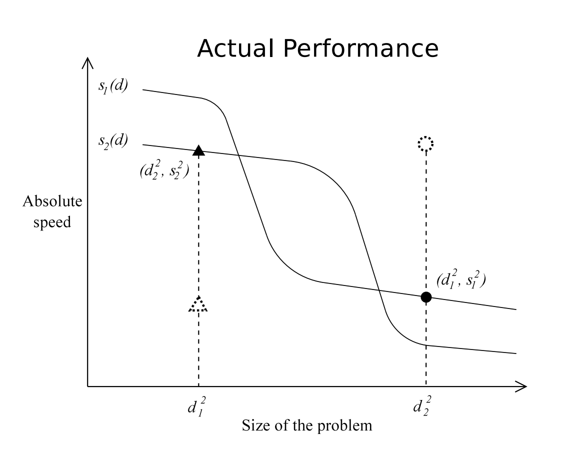
* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Straightforward dynamic load balancing may not balance



* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Straightforward dynamic load balancing may not balance



* Clarke, D. et al: Dynamic Load Balancing of Parallel Computational Iterative Routines on Highly Heterogeneous HPC Platforms. Parallel Processing Letters, 2011

Experimental Results

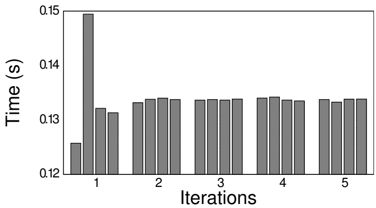
Iterative Routine

Jacobi method for solving a system of linear equations.

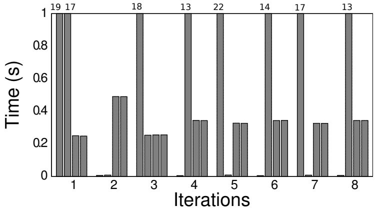
Experimental Setup

	P_1	P_2	P_3	P_4
Processor	3.6 Xeon	3.0 Xeon	3.4 P4	3.4 Xeon
Ram (MB)	256	256	512	1024

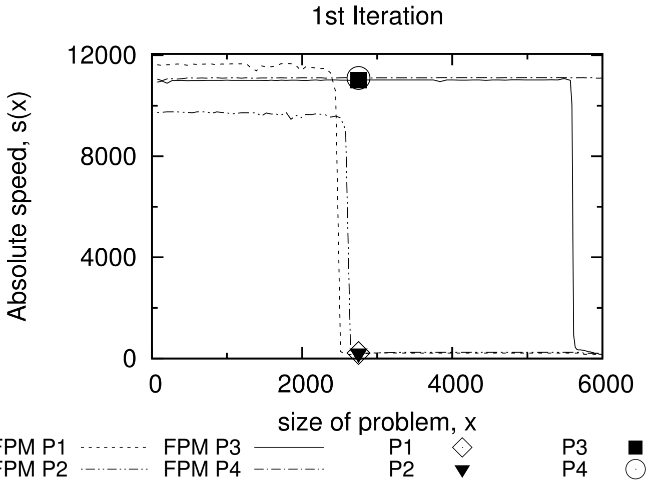
n = 8000



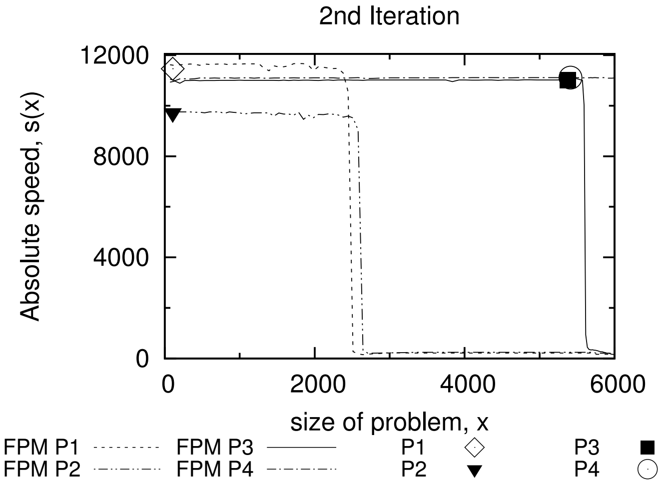
n = 11000



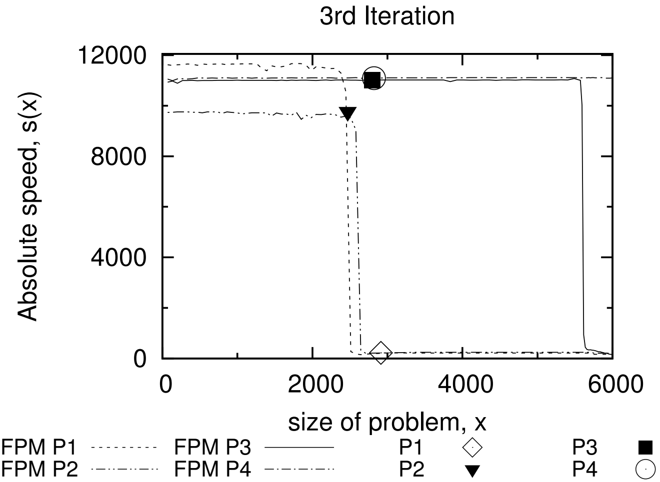
Experimental Results



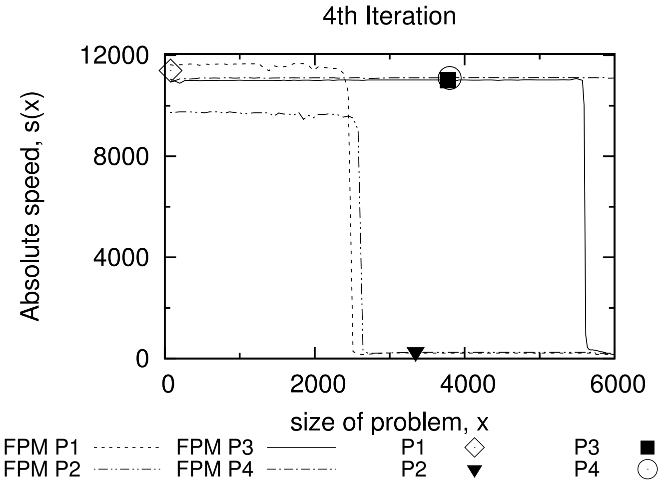
Experimental Results



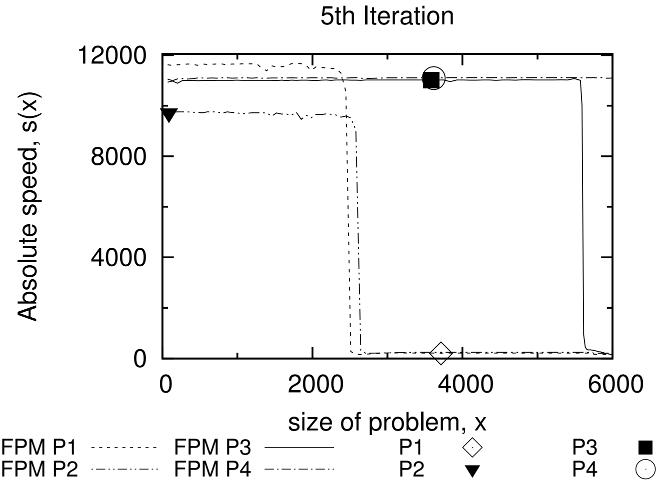
Experimental Results



Experimental Results



Experimental Results



Complex but realistic load balancing does always balance

Face the reality and use speed functions instead of constants in dynamic load balancing.

Challenges:

- Non-trivial partitioning algorithms manipulating by functions, not numbers.
- Non-trivial technique to build the speed functions suitable for the algorithms.

Benefits:

- Performance gains.

FPM-based Dynamic Load Balancing Algorithm

- Algorithm is based on models for which speed is a function of problem size.
- Load balancing achieved when:

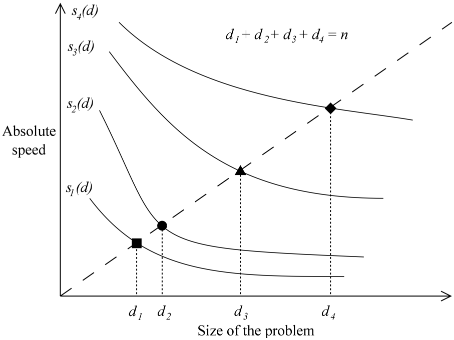
$$t_i \approx t_j, \quad 1 \leq i, j \leq p \tag{1}$$

$$\frac{d_1}{s_1(d_1)} \approx \frac{d_2}{s_2(d_2)} \approx \cdots \approx \frac{d_p}{s_p(d_p)} \tag{2}$$

where $d_1 + d_2 + \cdots + d_p = n$

Solving Distribution Problem

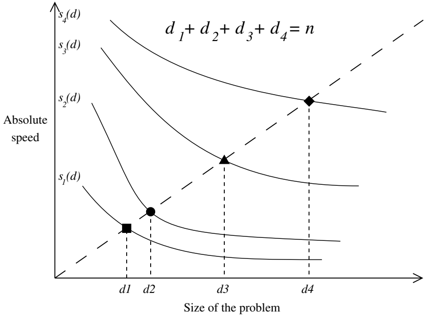
- Problem is solved geometrically by noting that the points $(d_i, s_i(d_i))$ lie on a line passing through the origin when $\frac{d_i}{s_i(d_i)} = \text{constant}$.



* Lastovetsky, A. and Reddy, R.: Data Partitioning with a Functional Performance Model of Heterogeneous Processors. Int. J. of High Perf. Comp. App., 2007

FPM-based data partitioning algorithm

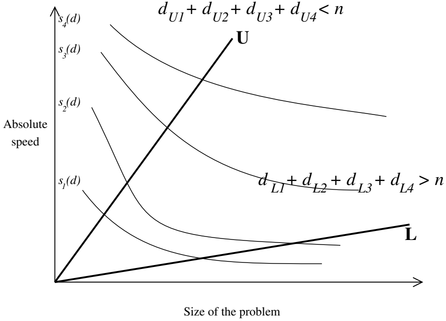
- Total problem size determines the slope
- Algorithm iteratively bisects solution space to find values d_i



* Lastovetsky, A. and Reddy, R.: Data Partitioning with a Functional Performance Model of Heterogeneous Processors. Int. J. of High Perf. Comp. App., 2007

FPM-based data partitioning algorithm

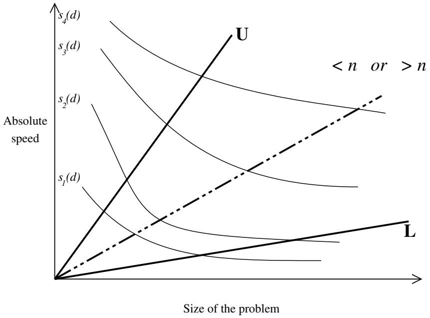
- Total problem size determines the slope
- Algorithm iteratively bisects solution space to find values d_i



* Lastovetsky, A. and Reddy, R.: Data Partitioning with a Functional Performance Model of Heterogeneous Processors. Int. J. of High Perf. Comp. App., 2007

FPM-based data partitioning algorithm

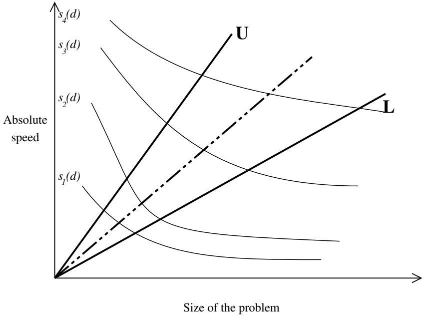
- Total problem size determines the slope
- Algorithm iteratively bisects solution space to find values d_i



* Lastovetsky, A. and Reddy, R.: Data Partitioning with a Functional Performance Model of Heterogeneous Processors. Int. J. of High Perf. Comp. App., 2007

FPM-based data partitioning algorithm

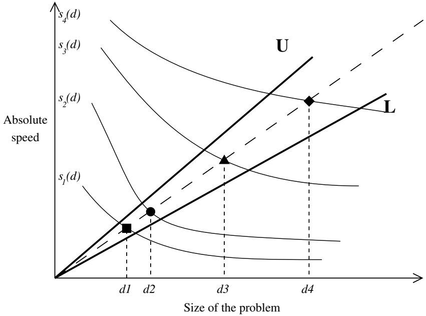
- Total problem size determines the slope
- Algorithm iteratively bisects solution space to find values d_i



* Lastovetsky, A. and Reddy, R.: Data Partitioning with a Functional Performance Model of Heterogeneous Processors. Int. J. of High Perf. Comp. App., 2007

FPM-based data partitioning algorithm

- Total problem size determines the slope
- Algorithm iteratively bisects solution space to find values d_i

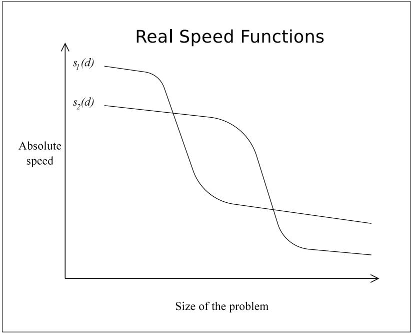


* Lastovetsky, A. and Reddy, R.: Data Partitioning with a Functional Performance Model of Heterogeneous Processors. Int. J. of High Perf. Comp. App., 2007

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

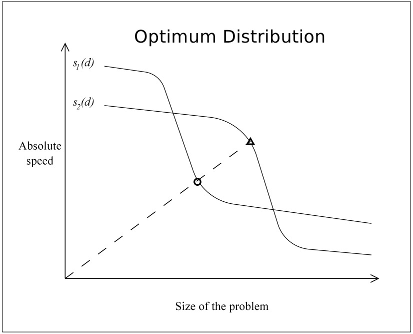


* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*



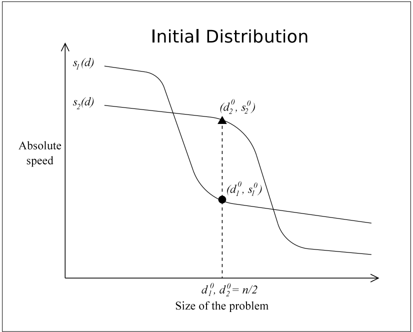
* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed
$$s_i^0 = \frac{n/p}{t_i(n/p)}$$
first function approximation
$$s_i'(x) \equiv s_i^0$$



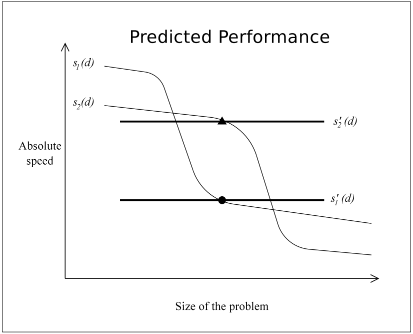
* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed
$$s_i^0 = \frac{n/p}{t_i(n/p)}$$
first function approximation
$$s_i'(x) \equiv s_i^0$$



* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed

$$s_i^0 = \frac{n/p}{t_i(n/p)}$$

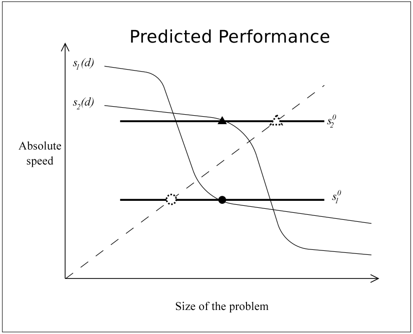
first function approximation

$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed

$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$

approximation $s_i'(x)$ updated by adding the point



* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed

$$s_i^0 = \frac{n/p}{t_i(n/p)}$$

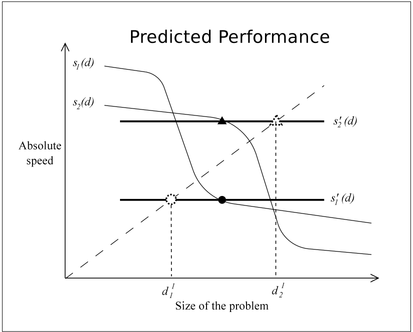
first function approximation

$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed

$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$

approximation $s_i'(x)$ updated by adding the point



* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed

$$s_i^0 = \frac{n/p}{t_i(n/p)}$$

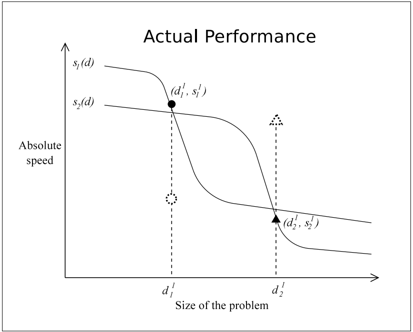
first function approximation

$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed

$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$

approximation $s_i'(x)$ updated by adding the point



* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

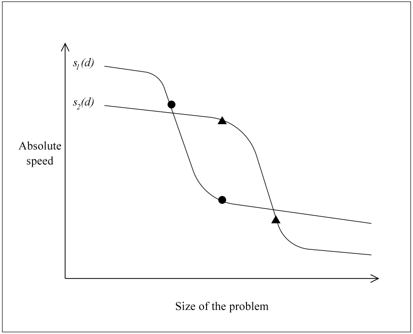
Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed
$$s_i^0 = \frac{n/p}{t_i(n/p)}$$
first function approximation
$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed
$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$
approximation $s_i'(x)$ updated by adding the point

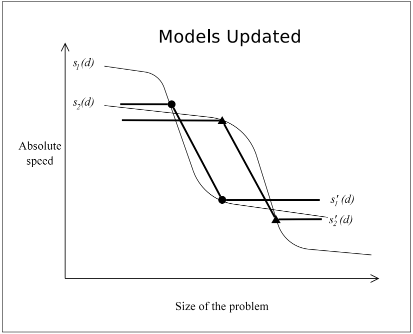


* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*



Initial: point $(n/p, s_i^0)$ with speed
$$s_i^0 = \frac{n/p}{t_i(n/p)}$$
first function approximation
$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed
$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$
approximation $s_i'(x)$ updated by adding the point

* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

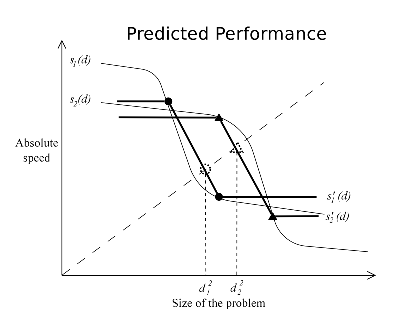
Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed
$$s_i^0 = \frac{n/p}{t_i(n/p)}$$
first function approximation
$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed
$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$
approximation $s_i'(x)$ updated by adding the point



* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

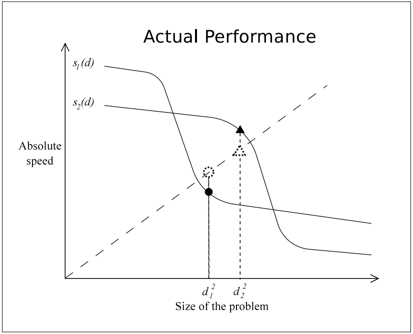
Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed
$$s_i^0 = \frac{n/p}{t_i(n/p)}$$
first function approximation
$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed
$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$
approximation $s_i'(x)$ updated by adding the point



* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

Initial: point $(n/p, s_i^0)$ with speed

$$s_i^0 = \frac{n/p}{t_i(n/p)}$$

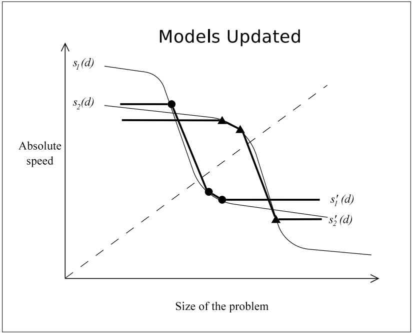
first function approximation

$$s_i'(x) \equiv s_i^0$$

Iterations: point (d_i^k, s_i^k) with speed

$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$

approximation $s_i'(x)$ updated by adding the point



* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

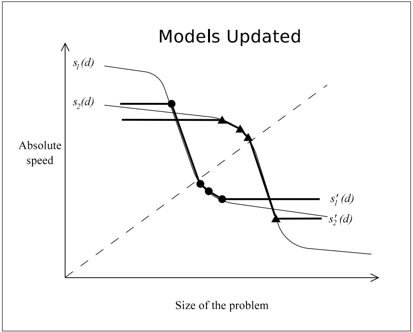
Dynamic FPM-based data partitioning

Functional Performance Models may be built:

- exhaustively in advance
- dynamically at run time*

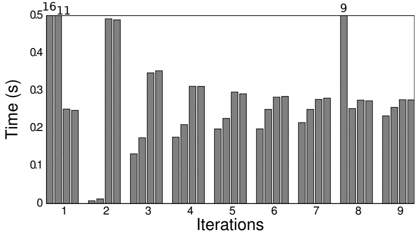
Initial: point $(n/p, s_i^0)$ with speed
$$s_i^0 = \frac{n/p}{t_i(n/p)}$$
first function approximation
$$s_i'(x) \equiv s_i^0$$

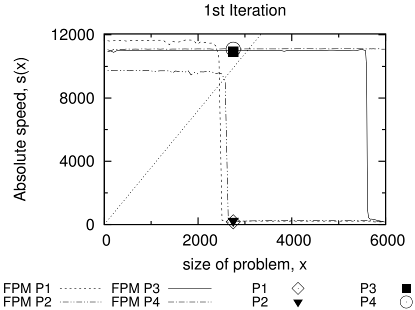
Iterations: point (d_i^k, s_i^k) with speed
$$s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$
approximation $s_i'(x)$ updated by adding the point



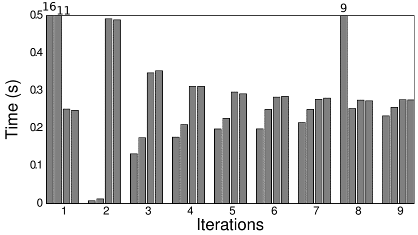
* Lastovetsky, A. and Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. HeteroPar'2009

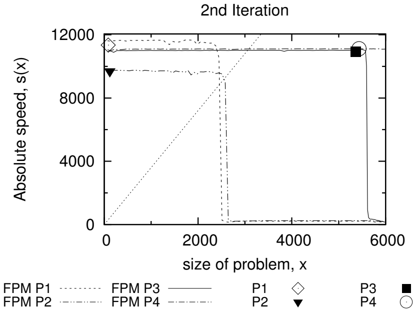
Experimental Results



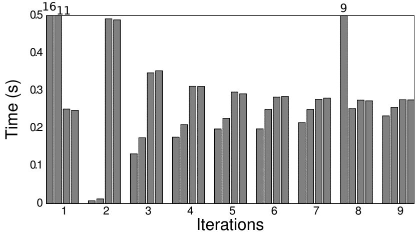


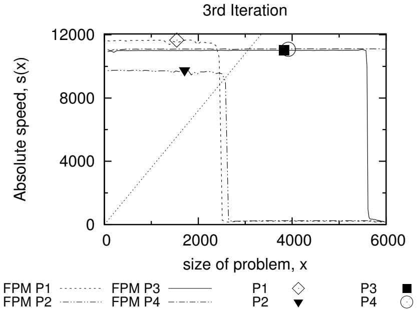
Experimental Results



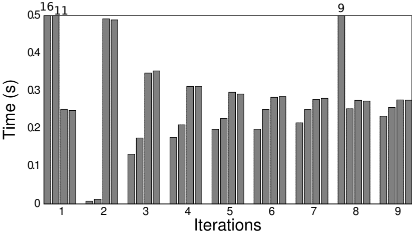


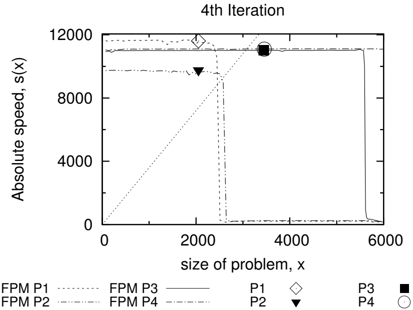
Experimental Results



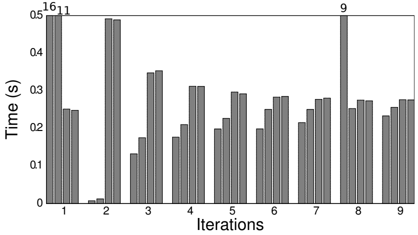


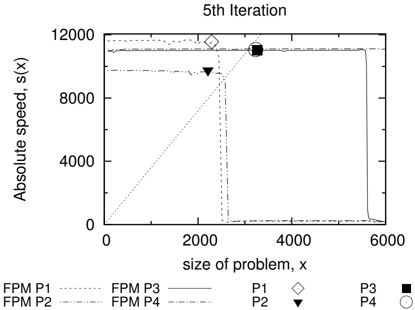
Experimental Results



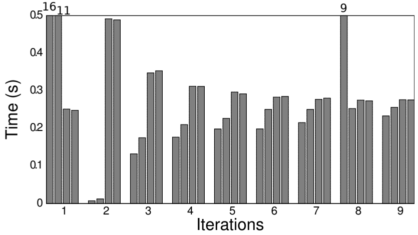


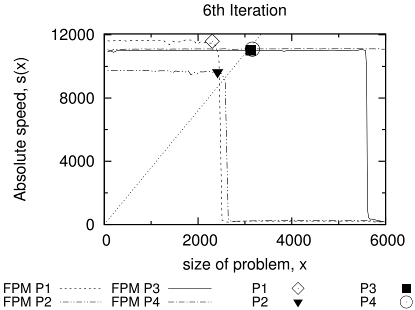
Experimental Results



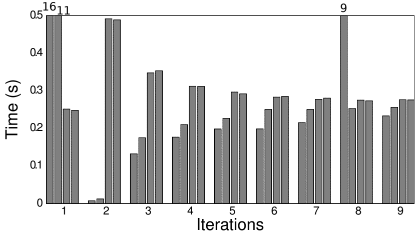


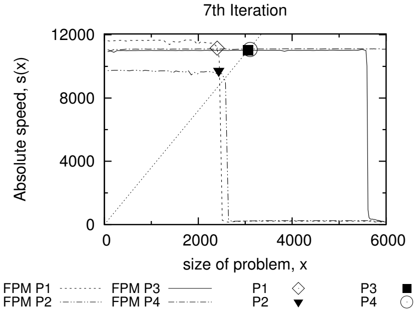
Experimental Results



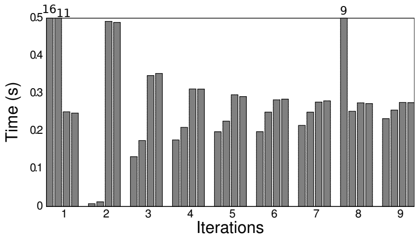


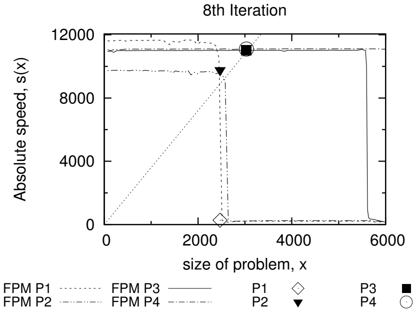
Experimental Results



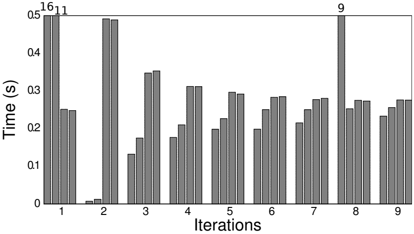


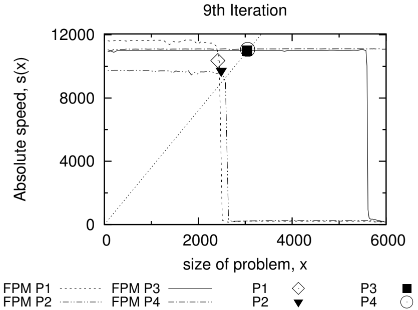
Experimental Results





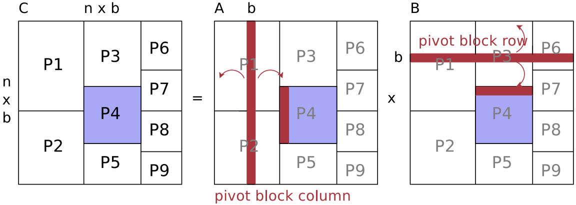
Experimental Results





Matrix Multiplication on Heterogeneous Platform*

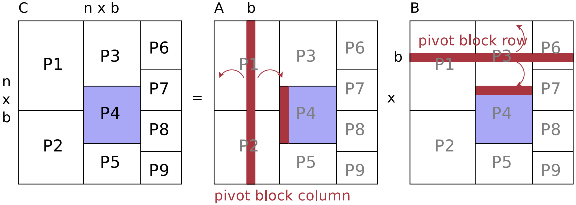
- Input: constant processor speeds
- Matrices partitioned so that
  - Area of the rectangle proportional to the speed
  - Volume of communication minimized



* Beaumont, O. et al: Matrix Multiplication on Heterogeneous Platforms. IEEE Trans. Parallel Distrib. Syst. 2001

Matrix Multiplication on Heterogeneous Platform*

- Input: constant processor speeds
- Matrices partitioned so that
  - Area of the rectangle proportional to the speed
  - Volume of communication minimized



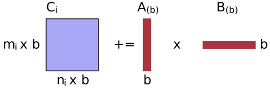
- More accurate solution is based on speed functions as input**

* Beaumont, O. et al: Matrix Multiplication on Heterogeneous Platforms. IEEE Trans. Parallel Distrib. Syst. 2001

** Clarke, D. et al: Column-Based Matrix Partitioning for Parallel Matrix Multiplication on Heterogeneous Processors Based on Functional Performance Models. In: HeteroPar-2011, LNCS 7155, 2012

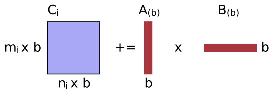
Matrix Multiplication on Heterogeneous Platform

- Computational kernel: panel-panel update

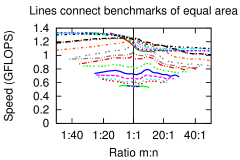


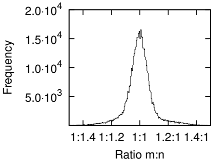
Matrix Multiplication on Heterogeneous Platform

- Computational kernel: panel-panel update



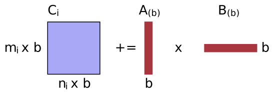
- Processor speed - function of area
  Built by running the kernel for square matrices



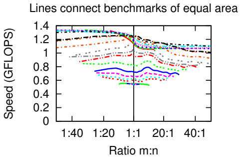


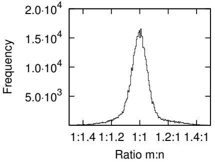
Matrix Multiplication on Heterogeneous Platform

- Computational kernel: panel-panel update

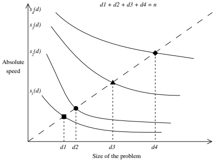


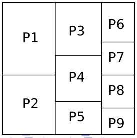
- Processor speed - function of area
  Built by running the kernel for square matrices



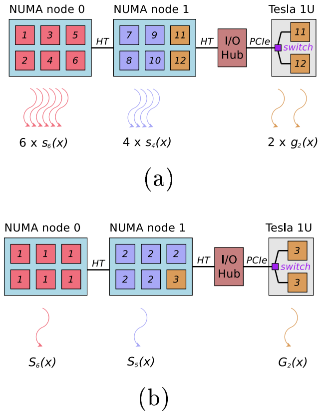


- FPM-based partitioning algorithm finds the optimal areas
  The areas are used as input to the matrix partitioning algorithm





Matrix multiply on hybrid node: performance modelling



* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

Matrix multiplication on hybrid node

Experimental platform

	CPU (AMD)	GPUs (NVIDIA)	
Architecture	Opteron 8439SE	GF GTX680	Tesla C870
Core Clock	2.8 GHz	1006 MHz	600 MHz
Number of Cores	4×6 cores	1536 cores	128 cores
Memory Size	4×16 GB	2048 MB	1536 MB
Memory Bandwidth		192.3 GB/s	76.8 GB/s

* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

Experiments on hybrid multicore multi-GPU node

Execution time of the application under different configurations

Matrix size (blks)	CPUs (sec)	GTX680 (sec)	Hybrid-FPM (sec)
40×40	99.5	74.2	26.6
50×50	195.4	162.7	77.8
60×60	300.1	316.8	114.4
70×70	491.6	554.8	226.1

Column 1: block size is 640×640
Column 2: 4×6 CPU cores, homogeneous data partitioning
Column 3: CPU core + GPU
Column 4: 2×6 CPU cores + 2×5 CPU cores + $2 \times$ (CPU core + GPU), FPM-based data partitioning ($12 \times s_6(x)$, $10 \times s_5(x)$, $2 \times g_2(x)$)

* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

Experiments on hybrid multicore multi-GPU node

Execution time of the application under different configurations

Matrix size (blks)	CPUs (sec)	GTX680 (sec)	Hybrid-FPM (sec)
40×40	99.5	74.2	26.6
50×50	195.4	162.7	77.8
60×60	300.1	316.8	114.4
70×70	491.6	554.8	226.1

Column 1: block size is 640×640
Column 2: 4×6 CPU cores, homogeneous data partitioning
Column 3: CPU core + GPU
Column 4: 2×6 CPU cores + 2×5 CPU cores + $2 \times$ (CPU core + GPU),
FPM-based data partitioning ($12 \times s_6(x)$, $10 \times s_5(x)$, $2 \times g_2(x)$)

* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

Experiments on hybrid multicore multi-GPU node

Execution time of the application under different configurations

Matrix size (blks)	CPUs (sec)	GTX680 (sec)	Hybrid-FPM (sec)
40×40	99.5	74.2	26.6
50×50	195.4	162.7	77.8
60×60	300.1	316.8	114.4
70×70	491.6	554.8	226.1

Column 1: block size is 640×640
Column 2: 4×6 CPU cores, homogeneous data partitioning
Column 3: CPU core + GPU
Column 4: 2×6 CPU cores + 2×5 CPU cores + $2 \times$ (CPU core + GPU),
FPM-based data partitioning ($12 \times s_6(x)$, $10 \times s_5(x)$, $2 \times g_2(x)$)

* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

Experiments on hybrid multicore multi-GPU node

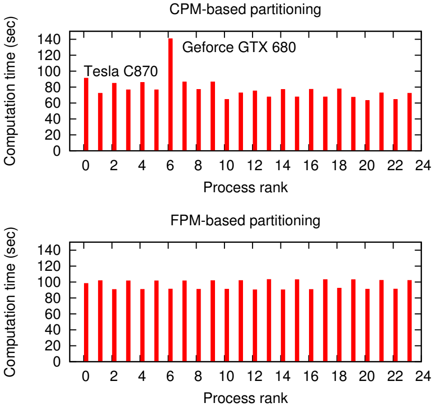
Execution time of the application under different configurations

Matrix size (blks)	CPUs (sec)	GTX680 (sec)	Hybrid-FPM (sec)
40×40	99.5	74.2	26.6
50×50	195.4	162.7	77.8
60×60	300.1	316.8	114.4
70×70	491.6	554.8	226.1

Column 1: block size is 640×640
Column 2: 4×6 CPU cores, homogeneous data partitioning
Column 3: CPU core + GPU
Column 4: 2×6 CPU cores + 2×5 CPU cores + $2 \times$ (CPU core + GPU), FPM-based data partitioning ($12 \times s_6(x)$, $10 \times s_5(x)$, $2 \times g_2(x)$)

* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

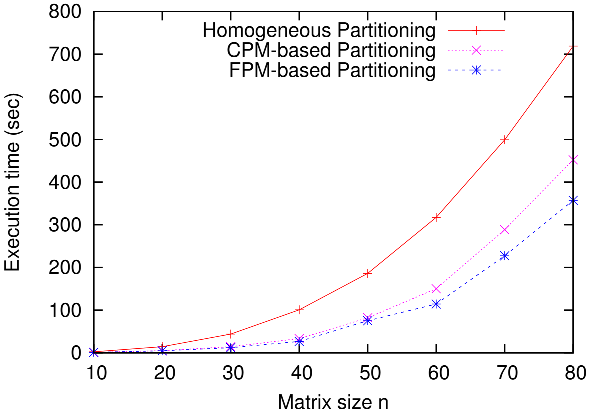
Computation time of each process



Matrix size 60×60, Computation time reduced by 40%

* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

Performance with different partitionings

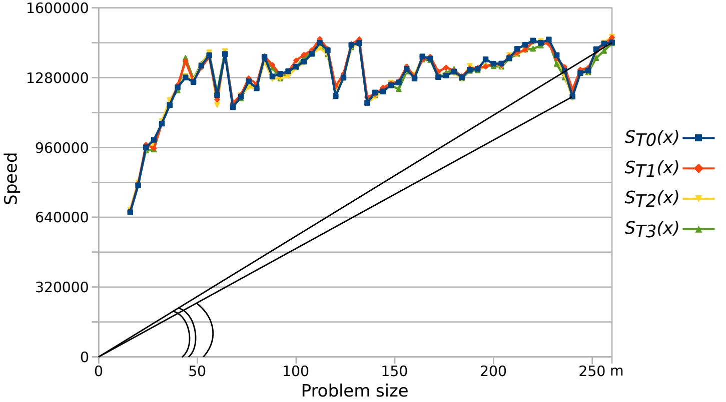


Execution time reduced by 23% and 45% respectively

* Zhong, Z. et al.: Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models. IEEE Transactions on Computers, 2015

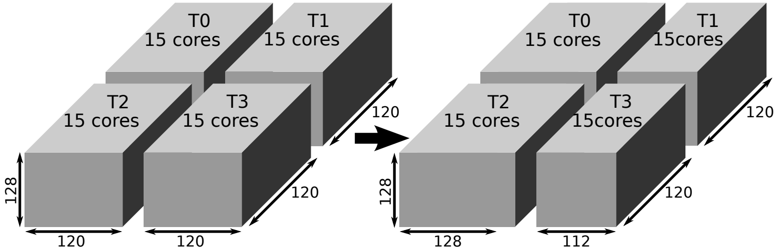
Performance optimization through load imbalancing

MPDATA can be optimized by imbalancing the load of processors*



* Lastovetsky, A., Shustak, L., Wyrzykowski, R.: Model-based optimization of MPDATA on Intel Xeon Phi through load imbalancing. arxiv.org preprint, 2015

Performance optimization through load imbalancing



Summary

- Traditional algorithms are easy to design but not accurate.
- FPM-based algorithms are not trivial and require mathematical skill beyond arithmetics and discrete maths.
  - This is a big problem and CS curriculum does not teach how to apply non-discrete maths in the context of CS. CS students believe that math analysis is something that only physicists need.
- It is not easy to go outside the box but benefits are there..

Спасибо за внимание!

Вопросы?