Dynamic Load Balancing of Parallel Computational Iterative Routines on Platforms with Memory Heterogeneity

David Clarke       Alexey Lastovetsky       Vladimir Rychkov

Heterogeneous Computing Laboratory
School of Computer Science and Informatics, University College Dublin,
Belfield, Dublin 4, Ireland
http://hcl.ucd.ie

HeteroPar’2010
Problem Outline

Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size
Outline

Problem Outline

Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm

Description of Algorithm
Analysis of Algorithm
Experimental Results
Outline

Problem Outline
   Iterative Routine
   Requirement for Load Balancing
   Speed as a function of problem size

Traditional Load Balancing Algorithm
   Description of Algorithm
   Analysis of Algorithm
   Experimental Results

Model Based Load Balancing Algorithm
   Description of Algorithm
   Analysis of Algorithm
   Experimental Results
Outline

Problem Outline
  Iterative Routine
  Requirement for Load Balancing
  Speed as a function of problem size

Traditional Load Balancing Algorithm
  Description of Algorithm
  Analysis of Algorithm
  Experimental Results

Model Based Load Balancing Algorithm
  Description of Algorithm
  Analysis of Algorithm
  Experimental Results

Conclusions
We present an algorithm for load balancing data-intensive parallel iterative routines.
We present an algorithm for load balancing data-intensive parallel iterative routines.

Target platform is a dedicated cluster with heterogeneous processors and heterogeneous distributed memory.
Problem Outline

Iterative Routine
- Requirement for Load Balancing
- Speed as a function of problem size

Traditional Load Balancing Algorithm
- Description of Algorithm
- Analysis of Algorithm
- Experimental Results

Model Based Load Balancing Algorithm
- Description of Algorithm
- Analysis of Algorithm
- Experimental Results

Conclusions
General Iterative Routine

\[ x^{k+1} = f(x^k) \quad k = 0, 1, ... \]  \hspace{1cm} (1)

\( x^k \) is an n-dimensional vector
\( f \) is some function from \( \mathbb{R}^n \) into itself.
General Iterative Routine

\[ x^{k+1} = f(x^k) \quad k = 0, 1, \ldots \quad (1) \]

\( x^k \) is an n-dimensional vector
\( f \) is some function from \( \mathbb{R}^n \) into itself.

Parallel Routine

- Data is partitioned over all processors
General Iterative Routine

\[ x^{k+1} = f(x^k) \quad k = 0, 1, \ldots \] (1)

\( x^k \) is an n-dimensional vector
\( f \) is some function from \( \mathbb{R}^n \) into itself.

Parallel Routine

- Data is partitioned over all processors
- Some independent calculations are carried out in parallel
General Iterative Routine

\[ x^{k+1} = f(x^k) \quad k = 0, 1, \ldots \]  \hspace{1cm} (1)

- \( x^k \) is an n-dimensional vector
- \( f \) is some function from \( \mathbb{R}^n \) into itself.

Parallel Routine

- Data is partitioned over all processors
- Some independent calculations are carried out in parallel
- Some data synchronisation takes place
General Iterative Routine

\[ x^{k+1} = f(x^k) \quad k = 0, 1, \ldots \]  (1)

\( x^k \) is an n-dimensional vector

\( f \) is some function from \( \mathbb{R}^n \) into itself.

Parallel Routine

- Data is partitioned over all processors
- Some independent calculations are carried out in parallel
- Some data synchronisation takes place

Typically computational workload is directly proportional to the size of data
Problem Outline

Iterative Routine

Requirement for Load Balancing

Speed as a function of problem size

Traditional Load Balancing Algorithm

Description of Algorithm

Analysis of Algorithm

Experimental Results

Model Based Load Balancing Algorithm

Description of Algorithm

Analysis of Algorithm

Experimental Results

Conclusions
Load balancing minimises overall computation time.
Load balancing minimises overall computation time.

All processors should complete an iteration in the same time.
Load balancing minimises overall computation time.

All processors should complete an iteration in the same time.
Load balancing minimises overall computation time.

All processors should complete an iteration in the same time.

On a heterogeneous cluster this is achieved by partitioning data and calculations in proportion to processor speed.
Problem Outline

Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Model Based Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Conclusions
Traditionally, processor performance is defined by a constant number.

In reality, speed is a function of problem size. Algorithms based on constant performance models are only applicable for limited problem sizes.
Speed as a function of problem size

- Traditionally, processor performance is defined by a constant number.
- In reality, speed is a function of problem size.
Traditionally, processor performance is defined by a constant number.

In reality, speed is a function of problem size.
Traditionally, processor performance is defined by a constant number.

In reality, speed is a function of problem size.

Algorithms based on constant performance models are only applicable for limited problem sizes.
Problem Outline

Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Model Based Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Conclusions
Traditional Load Balancing Algorithms

\( n \) computational units distributed across \( p \) processors.
Traditional Load Balancing Algorithms

$n$ computational units distributed across $p$ processors. Processor $P_i$ has $d_i$ units such that $n = \sum_{i=1}^{p} d_i$
Traditional Load Balancing Algorithms

$n$ computational units distributed across $p$ processors. Processor $P_i$ has $d_i$ units such that $n = \sum_{i=1}^p d_i$

Initially $d_i^0 = n/p$
Traditional Load Balancing Algorithms

\( n \) 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 measured and gathered to root
Traditional Load Balancing Algorithms

$n$ 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 measured and gathered to root

2. if relative difference between times $\leq \epsilon$
   
   then no balancing needed
Traditional Load Balancing Algorithms

\( n \) 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 measured and gathered to root
2. if relative difference between times \( \leq \epsilon \) then no balancing needed
   else new distribution is calculated as:
   \[
   d_{i}^{k+1} = n \times \frac{s_{i}^{k}}{\sum_{j=1}^{p} s_{j}^{k}} \text{ where speed } s_{i}^{k} = \frac{d_{i}^{k}}{t_{i}(d_{i}^{k})}
   \]
Traditional Load Balancing Algorithms

$n$ 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 measured and gathered to root
2. if relative difference between times $\leq \epsilon$ then no balancing needed
   else new distribution is calculated as:
   $$d_i^{k+1} = n \times \frac{s_i^k}{\sum_{j=1}^{p} s_j^k} \text{ where speed } s_i^k = \frac{d_i^k}{t_i(d_i^k)}$$
3. New distributions $d_i^{k+1}$ broadcast to all processors and where necessary data is redistributed accordingly.
Dynamic Load Balancing of Iterative Routines

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Problem Outline
Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Model Based Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Conclusions
Problem Outline

Traditional Load Balancing Algorithm

Model Based Load Balancing Algorithm

Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results

Traditional Load Balancing Algorithm

- Speed of each processor is considered as a constant positive number at each iteration.
Traditional Load Balancing Algorithm

- Speed of each processor is considered as a constant positive number at each iteration.
- Within the range of problem sizes for which this is true, traditional algorithms can successfully load balance.
Traditional Load Balancing Algorithm

- Speed of each processor is considered as a constant positive number at each iteration.
- Within the range of problem sizes for which this is true, traditional algorithms can successfully load balance.
- Can fail for problem sizes for which the speed is not constant.
Real Speed Functions

$s_1(d)$

$s_2(d)$

Absolute speed

Size of the problem
Dynamic Load Balancing of Iterative Routines

Initial Distribution

Absolute speed

\[ s_1(d) \]

\[ s_2(d) \]

\( (d_2^0, s_2^0) \)

\( (d_1^0, s_1^0) \)

\[ d_1^0, d_2^0 = n/2 \]

Size of the problem
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

Predicted Performance

$s_1(d)$
$s_2(d)$
$s_2^0$
$s_1^0$

Absolute speed
Size of the problem
Problem Outline
Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Dynamic Load Balancing of Iterative Routines
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

Dynamic Load Balancing of Iterative Routines
Problem Outline

Traditional Load Balancing Algorithm

Model Based Load Balancing Algorithm

Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

Predicted Performance

$s_1(d)$

$s_2(d)$

$s_i^j$

Absolute speed

Size of the problem
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

Predicted Performance

\[ s_1(d) \]
\[ s_2(d) \]

Absolute speed

Size of the problem
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

Dynamic Load Balancing of Iterative Routines
Actual Performance

- $s_1(d)$
- $s_2(d)$

$(d_2^2, s_2^2)$

$(d_1^2, s_1^2)$

Absolute speed

Size of the problem

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Layout

Problem Outline
Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Model Based Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Conclusions
Iterative Routine

Jacobi method for solving a system of linear equations.
Iterative Routine

Jacobi method for solving a system of linear equations.

Experimental Setup

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>3.6 Xeon</td>
<td>3.0 Xeon</td>
<td>3.4 P4</td>
<td>3.4 Xeon</td>
</tr>
<tr>
<td>Ram (MB)</td>
<td>256</td>
<td>256</td>
<td>512</td>
<td>1024</td>
</tr>
</tbody>
</table>
Iterative Routine

Jacobi method for solving a system of linear equations.

Experimental Setup

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>3.6 Xeon</td>
<td>3.0 Xeon</td>
<td>3.4 P4</td>
<td>3.4 Xeon</td>
</tr>
<tr>
<td>Ram (MB)</td>
<td>256</td>
<td>256</td>
<td>512</td>
<td>1024</td>
</tr>
</tbody>
</table>

$n = 8000$
Iterative Routine

Jacobi method for solving a system of linear equations.

Experimental Setup

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>3.6 Xeon</td>
<td>3.0 Xeon</td>
<td>3.4 P4</td>
<td>3.4 Xeon</td>
</tr>
<tr>
<td>Ram (MB)</td>
<td>256</td>
<td>256</td>
<td>512</td>
<td>1024</td>
</tr>
</tbody>
</table>

$n = 8000$

![Graph for $n = 8000$](image1)

$n = 11000$

![Graph for $n = 11000$](image2)
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Dynamic Load Balancing of Iterative Routines
Problem Outline

- Traditional Load Balancing Algorithm
- Model Based Load Balancing Algorithm
- Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Dynamic Load Balancing of Iterative Routines
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Dynamic Load Balancing of Iterative Routines
Problem Outline

Traditional Load Balancing Algorithm
Model Based Load Balancing Algorithm
Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Dynamic Load Balancing of Iterative Routines
Problem Outline

Traditional Load Balancing Algorithm

Model Based Load Balancing Algorithm

Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Layout

Problem Outline
Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Model Based Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Conclusions
Our algorithm is based on models for which speed is a function of problem size.
Our algorithm is based on models for which speed is a function of problem size.

Load balancing achieved when:

\[ t_i \approx t_j, \quad 1 \leq i, j \leq p \]  

\[ \frac{d_1}{s_1(d_1)} \approx \frac{d_2}{s_2(d_2)} \approx \ldots \approx \frac{d_p}{s_p(d_p)} \]  

where \( d_1 + d_2 + \cdots + d_p = n \)
Problem is solved geometrically by noting that the points \((d_i, s_i(d_i))\) lie on a line passing through the origin when \(\frac{d_i}{s_i(d_i)} = constant\).
Our Dynamic Load Balancing Algorithm

- These functional performance models are different for each routine on each processor.
Our Dynamic Load Balancing Algorithm

- These functional performance models are different for each routine on each processor.
- Building these models for all conceivable problem sizes is very computationally expensive.
These functional performance models are different for each routine on each processor.

Building these models for all conceivable problem sizes is very computationally expensive.

Building full models is not an option for a self adaptive algorithm.
Our Dynamic Load Balancing Algorithm

- These functional performance models are different for each routine on each processor.
- Building these models for all conceivable problem sizes is very computationally expensive.
- Building full models is not an option for a self adaptive algorithm.
- Our algorithm dynamically builds the models at relevant problem sizes using piecewise linear approximations.
First iteration Point \((\frac{n}{p}, s_i^0)\) with speed \(s_i^0 = \frac{n/p}{t_i(n/p)}\)
First iteration  
Point \( \left( \frac{n}{p}, s_i^0 \right) \) with speed 
\[
s_i^0 = \frac{n/p}{t_i(n/p)}
\]

First function approximation \( s'_i(d) = s_i^0 \)
First iteration Point \( (\frac{n}{p}, s^0_i) \) with speed \( s^0_i = \frac{n/p}{t_i(n/p)} \)

First function approximation \( s'_i(d) = s^0_i \)

Subsequent iterations Point \((d_i^k, s^k_i)\) with speed \( s^k_i = \frac{d_i^k}{t_i(d_i^k)} \)
First iteration  Point \((\frac{n}{p}, s^0_i)\) with speed \(s^0_i = \frac{n/p}{t_i(n/p)}\)
First function approximation \(s'_i(d) = s^0_i\)

Subsequent iterations  Point \((d^k_i, s^k_i)\) with speed \(s^k_i = \frac{d^k_i}{t_i(d^k_i)}\)
Function approximation updated by adding the point
First iteration  Point \( \left( \frac{n}{p}, s_i^0 \right) \) with speed \( s_i^0 = \frac{n/p}{t_i(n/p)} \)

First function approximation \( s_i'(d) = s_i^0 \)

Subsequent iterations  Point \( (d_i^k, s_i^k) \) with speed \( s_i^k = \frac{d_i^k}{t_i(d_i^k)} \)

Function approximation updated by adding the point
First iteration  \( \left( \frac{n}{p}, s_i^0 \right) \) with speed \( s_i^0 = \frac{n/p}{t_i(n/p)} \)
First function approximation \( s'_i(d) = s_i^0 \)

Subsequent iterations  \( \left( d_i^k, s_i^k \right) \) with speed \( s_i^k = \frac{d_i^k}{t_i(d_i^k)} \)
Function approximation updated by adding the point
First iteration  Point \( \left( \frac{n}{p}, s_i^0 \right) \) with speed \( s_i^0 = \frac{n/p}{t_i(n/p)} \)
First function approximation \( s_i'(d) = s_i^0 \)

Subsequent iterations  Point \( (d_i^k, s_i^k) \) with speed \( s_i^k = \frac{d_i^k}{t_i(d_i^k)} \)
Function approximation updated by adding the point
First iteration
Point \( \left( \frac{n}{p}, s_i^0 \right) \) with speed \( s_i^0 = \frac{n/p}{t_i(n/p)} \)

First function approximation \( s'_i(d) = s_i^0 \)

Subsequent iterations
Point \( (d_i^k, s_i^k) \) with speed \( s_i^k = \frac{d_i^k}{t_i(d_i^k)} \)

Function approximation updated by adding the point
Problem Outline
Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Model Based Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Conclusions
Real Speed Functions

$s_1(d)$

$s_2(d)$

Absolute speed

Size of the problem
Problem Outline
- Traditional Load Balancing Algorithm
- Model Based Load Balancing Algorithm
- Conclusions

Description of Algorithm
Analysis of Algorithm
Experimental Results

Optimum Distribution

$s_1(d)$
$s_2(d)$

Absolute speed

Size of the problem

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Dynamic Load Balancing of Iterative Routines
Initial Distribution

\( s_1(d) \)
\( s_2(d) \)

\((d_2^0, s_2^0)\)

\((d_1^0, s_1^0)\)

\( d_1^0, d_2^0 = n/2 \)

Size of the problem
Predicted Performance

$s_1(d)$

$s_2(d)$

$s_1'(d)$

$s_2'(d)$

Absolute speed

Size of the problem

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Predicted Performance

Absolute speed

Size of the problem

$s_1(d)$

$s_2(d)$

$s_1^0$

$s_2^0$
Predicted Performance

\[ s_1(d) \]
\[ s_2(d) \]

Absolute speed

Size of the problem

\[ d_1 \]
\[ d_2 \]
Actual Performance

- $s_1(d)$
- $s_2(d)$

Absolute speed

Size of the problem

$(d_1^l, s_1^l)$

$(d_2^l, s_2^l)$
Problem Outline
- Traditional Load Balancing Algorithm
- Model Based Load Balancing Algorithm
- Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Models Updated

$s_1(d)$

$s_2(d)$

$s_1'(d)$

$s_2'(d)$

Absolute speed

Size of the problem
Problem Outline

- Traditional Load Balancing Algorithm
- Model Based Load Balancing Algorithm
- Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Actual Performance

$s_1(d)$
$s_2(d)$

Absolute speed

Size of the problem $d_1^2$ $d_2^2$
Problem Outline

Traditional Load Balancing Algorithm

Model Based Load Balancing Algorithm

Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results
Dynamic Load Balancing of Iterative Routines

- Problem Outline
  - Traditional Load Balancing Algorithm
  - Model Based Load Balancing Algorithm
  - Conclusions

- Description of Algorithm
- Analysis of Algorithm
- Experimental Results

Models Updated

\[ s_1(d), s_2(d), s_1'(d), s_2'(d) \]

Absolute speed

Size of the problem
Problem Outline
Iterative Routine
Requirement for Load Balancing
Speed as a function of problem size

Traditional Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Model Based Load Balancing Algorithm
Description of Algorithm
Analysis of Algorithm
Experimental Results

Conclusions
Experimental Results
Experimental Results

![Graph showing time (s) vs. iterations for different problem sizes. The graph includes data for FPM P1, FPM P2, FPM P3, and FPM P4.]
Experimental Results

- Time (s) vs. Iterations
- Absolute speed, s(x) vs. size of problem, x
- 4th Iteration

Graphs showing performance metrics for load balancing algorithms.
Experimental Results

- Bar chart showing time (s) vs. iterations for different iterations.
- Line graph showing absolute speed vs. size of problem for 5th iteration.

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Experimental Results

- Time (s) vs. Iterations
- Absolute speed, s(x) vs. size of problem, x
- Graphs showing the performance of FPM P1 to P4 and P1 to P4 over iterations

David Clarke, Alexey Lastovetsky, Vladimir Rychkov
Dynamic Load Balancing of Iterative Routines
Experimental Results

![Graph 1: Time vs. Iterations](image1.png)

![Graph 2: Absolute speed vs. size of problem](image2.png)

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Problem Outline

Traditional Load Balancing Algorithm

Model Based Load Balancing Algorithm

Conclusions

Description of Algorithm

Analysis of Algorithm

Experimental Results

Experimental Results

![Graph showing time vs iterations for different iterations.](image1)

![Graph showing absolute speed vs size of problem for 8th iteration.](image2)

David Clarke, Alexey Lastovetsky, Vladimir Rychkov

Dynamic Load Balancing of Iterative Routines
Experimental Results

- Time (s) vs. Iterations
- Absolute speed, s(x) vs. size of problem, x

FPM P1, FPM P2, FPM P3, FPM P4, P1, P2, P3, P4
Conclusions

- Traditional algorithms only work for problems which fit into the main memory of all processors.
Conclusions

- Traditional algorithms only work for problems which fit into the main memory of all processors.
- Our algorithm, based on functional performance models, can balance for all problem sizes.
Conclusions

- Traditional algorithms only work for problems which fit into the main memory of all processors.
- Our algorithm, based on functional performance models, can balance for all problem sizes.
- No prior information about the heterogeneity and memory hierarchy of the platform needed as inputs into the algorithm.
Conclusions

- Traditional algorithms only work for problems which fit into the main memory of all processors.
- Our algorithm, based on functional performance models, can balance for all problem sizes.
- No prior information about the heterogeneity and memory hierarchy of the platform needed as inputs into the algorithm.
- Can be deployed self adaptively on any dedicated platform.
Questions?