Static and Dynamic LoadBalancing for Heterogeneous Systems

Greg Michaelson
Turkey Al Salkini, Khari Armih, Deng Xiao Yan
Heriot-Watt University
Phil Trinder
University of Glasgow
Context

• explosive growth in deployment of heterogeneous systems
  – desktop/laptop/tablet with many-core + GPU
• HPC with clusters of nodes of many-core + GPU
• upgrade/replace/remove cluster components as opportune rather than replacing whole cluster
• newer components typically higher spec than older components
• hard to allocate and balance load
Context

- typical heterogeneous system
- colours indicate different technologies
Context

- static work allocation for regular data parallelism
- simple data parallel skeleton with master & identical workers
- masters sends work to workers and receives results
- assume: master on core; workers on cores or GPU
Homogeneity

- cluster of identical single core nodes with no GPU
Amdahl’s Law

- $T = \text{total sequential time}$
Amdahl’s Law

• $T = \text{total sequential time}$
• $T_{\text{seq}} = \text{necessarily sequential time}$
• $T_{\text{par}} = \text{potentially parallelisable time}$
• i.e. $T = T_{\text{seq}} + T_{\text{par}}$
Amdahl’s Law

- $T = \text{total sequential time}$
- $T_{seq} = \text{necessarily sequential time}$
- $T_{par} = \text{potentially parallelisable time}$
- i.e. $T = T_{seq} + T_{par}$
- $N = \text{number of nodes}$
- allocate same amounts of $T_{par}$ to each node
Amdahl’s Law

• $T = \text{total sequential time}$
• $T_{seq} = \text{necessarily sequential time}$
• $T_{par} = \text{potentially parallelisable time}$
• i.e. $T = T_{seq} + T_{par}$
• $N = \text{number of nodes}$
• allocate same amounts of $T_{par}$ to each node
• $T_{min} = \text{minimum time}$
• $T_{min} = T_{seq} + T_{par}/N$
Communication cost

- $T_{send}$ = time to send data to processors
- $T_{rec}$ = time to receive results from processors
- for any parallel benefit, need:
  - $T_{send} + T_{rec} + T_{par}/N < T_{par}$
Heterogeneity 1

- different number of cores/amount of RAM on each node
- same core/RAM technologies
Heterogeneity 1

• $Cores_i =$ number of cores on node $i$
• $P =$ number of processors = $Cores_1 + ... + Cores_N$
• each core gets: $T_{par}/P$
• node $i$ gets: $Cores_i * T_{par}/P$
• haven’t taken memory into account...?
Heterogeneity 2

• different technologies on each node
• no GPU
Heterogeneity 2

• characterise node strength
  – coarse-grain, relative measure
• for node $i$:
  $Speed_i = $ clock speed of each core
  $Strength_i = Cores_i * Speed_i$
  i.e. strength increases with number of cores & core speed
• whole system: $Strength = Strength_1 + ... + Strength_N$
• node $i$ gets: $T_{par} * Strength_i / Strength$
• each node $i$ core gets: $(T_{par} * Strength_i / Strength) / Cores_i$
Heterogeneity 2

• still doesn’t take memory into account?
• top level shared cache (i.e. L2 or L3) most important
  – determines frequencies of access clashes & caching
• for node $i$:

  $\text{Cache}_i = \text{shared cache size}$

  $\text{Strength}_i = \text{Cores}_i \cdot \text{Speed}_i \cdot \text{Cache}_i$

  i.e. strength also increases with cache size
• use previous equations for system strength, and node and core load
### Heterogeneity 2

<table>
<thead>
<tr>
<th>Machine name</th>
<th>Architecture</th>
<th>CPU</th>
<th>Case1</th>
<th>Case2</th>
<th>Case3</th>
</tr>
</thead>
<tbody>
<tr>
<td>lxparal</td>
<td>Xeon 4510</td>
<td>8</td>
<td>1998</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>lxparal2</td>
<td>Xeon 4510</td>
<td>8</td>
<td>1998</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>brahma</td>
<td>Xeon(TM)</td>
<td>4</td>
<td>3065</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>amaterasu</td>
<td>Core(TM) i7</td>
<td>4</td>
<td>1199</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>jove</td>
<td>Core(TM)i7</td>
<td>8</td>
<td>1200</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>linux01</td>
<td>2 Duo CPU</td>
<td>2</td>
<td>1200</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>linux02</td>
<td>2 Duo CPU</td>
<td>2</td>
<td>1200</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>linux03</td>
<td>2 Duo CPU</td>
<td>2</td>
<td>1200</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>linux04</td>
<td>2 Duo CPU</td>
<td>2</td>
<td>1200</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Heterogeneity 2

**Sum-Euler (Case1)**

- Without Cost Model
- With Cost Model
- Predicted Max Speedup

<table>
<thead>
<tr>
<th>Cores</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>28</td>
<td>25</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

1-8 lxpara, 9-12 brahma, 13-16 amaterasu, 17-24 jove, 25-30 linux
(2g/6mb) (3g/512kb) (1g/8mb) (1g/8mb) (1g/2mb)

**Image Matching (Case1)**

- Without Cost Model
- With Cost Model
- Predicted Max Speedup

<table>
<thead>
<tr>
<th>Cores</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>28</td>
<td>25</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

1-8 lxpara, 9-12 brahma, 13-16 amaterasu, 17-24 jove, 25-30 linux
(2g/6mb) (3g/512kb) (1g/8mb) (1g/8mb) (1g/2mb)
Heterogeneity 3

- different node technologies
- with GPU
Heterogeneity 3

• wish to treat GPU as additional general compute resource - \textit{GP-GPU}
• but cannot directly adopt current algorithm independent model
  – very hard to characterise GP-GPU succinctly from static information
• base GPU strength measure on algorithm specific profiling
• use cost model for core strength
• normalise strength relative to one core
Heterogeneity 3

\[ T_{\text{Core}}_0 = \text{time for algorithm on base core} \]

- for node i:

\[ T_{\text{GPU}}_i = \text{time for algorithm on GPU} \]

- strength of \( GPU_i = \frac{T_{\text{GPU}}_i}{T_{\text{Core}}_0} \)

- node i core strength = \( \frac{\text{Speed}_i \times \text{Cache}_i}{\text{Speed}_0 \times \text{Cache}_0} \)

- assume GPU controlled by one core

\[ \text{Strength}_i = \frac{T_{\text{GPU}}_i}{T_{\text{Core}}_0} + (\text{Core}_i - 1) \times \frac{\text{Speed}_i \times \text{Cache}_i}{\text{Speed}_0 \times \text{Cache}_0} \]
Heterogeneity 3

Fibonacci (linux_lab)

Fibonacci (lxpara)
Heterogeneity 3

Matrix Multiplication (linux_lab)

Matrix Multiplication (lxpara)
Heterogeneity 3

Fibonacci 150,000

Architectures

Speed up

Cores
GPU
GPU + Cores-1
Task mobility

- static model assumes program has sole use of system
  - no load changes at run-time
- if system shared then load changes unpredictably at run-time
- base system design on threads?
- hope operating system will manage load?
  - optimising movement of arbitrary threads across nodes is a black art
- decentralise mobility
- let master decide when to move work
Task mobility

M: Master, \( W_i \): Worker, Colour: Amount of load on processor.
W3 becomes overloaded. The load should be balanced by moving the work to a lightly loaded processor.
Task mobility

The load is re-balanced.
Mobility decision

\( T_i = \) time to complete task on location \( i \)

\( T_{move} = \) time to move task from current to new location

\( Overhead = \) acceptable completion overhead e.g. 5%

- to justify moving work from location \( i \) to location \( j \):

\[ T_j + T_{move} < T_i \times Overhead \]
Mobility decision

• homogeneous case

\( T_{Base} = \text{time to complete whole task on unloaded core} \)

\( \text{Load}_{i,t} = \text{load on location } i \text{ at time } t \)

• assume we inspect load having completed \( P\% \) of task

\( T_i = T_{Base} \frac{(100-P)}{100} f(\text{Load}_{i,t}, \text{Load}_{i,0}) \)
Evaluation

- 7,000*7,000 matrix multiplication
- Poisson load
- darker == more heavily loaded

16 workers + 10 tasks, no mobility
16 workers + 10 tasks, mobility
Conclusions

• simple models + relative measures suffice for static & dynamic load balancing
• data parallel/regular parallelism only
• small scale experiments
• next:
  – larger scale experiments
  – explore more complex algorithms
  – extend mobility to many core
References

• T. Al Salkini and G. Michaelson, ‘Dynamic Farm Skeleton Task Allocation Through Task Mobility’, 18th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'12), Las Vegas, July 16-19, 2012