BSC Tools update
Using clustering and Folding

Judit Gimenez (judit@bsc.es)
Outline

- Computation Structure detection
  - Short intro
  - Aggregative Refinement
  - Tracking program evolution
  - Scaling clustering algorithm

- Instantaneous performance metric
  - Clustering + Folding
Clustering

Identification of computation structure
- CPU burst = region between consecutive runtime calls
  - Described with performance hardware counters
  - Associated with call stack data

Using DBSCAN density-cluster algorithm
- Data not necessarily Gaussian
Outputs

Scatter Plot of Clustering Metrics

Clusters Distribution Along Time

Cluster Statistics

<table>
<thead>
<tr>
<th>CLUSTER</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>% TIME</td>
<td>36.29</td>
<td>29.52</td>
<td>10.13</td>
<td>9.68</td>
<td>3.73</td>
<td>1.71</td>
</tr>
<tr>
<td>AVG. BURST DUR. (MS)</td>
<td>220.46</td>
<td>177.70</td>
<td>60.81</td>
<td>29.09</td>
<td>38.71</td>
<td>44.83</td>
</tr>
<tr>
<td>IPC</td>
<td>0.53</td>
<td>0.50</td>
<td>0.62</td>
<td>0.77</td>
<td>0.66</td>
<td>0.59</td>
</tr>
<tr>
<td>MIPS</td>
<td>1210.07</td>
<td>1164.36</td>
<td>1403.19</td>
<td>1743.32</td>
<td>1499.47</td>
<td>1338.24</td>
</tr>
<tr>
<td>L1M/KINSTR</td>
<td>22.72</td>
<td>32.63</td>
<td>12.65</td>
<td>8.39</td>
<td>16.12</td>
<td>6.86</td>
</tr>
<tr>
<td>L2M/KINSTR</td>
<td>0.59</td>
<td>1.23</td>
<td>1.08</td>
<td>0.61</td>
<td>1.23</td>
<td>1.73</td>
</tr>
<tr>
<td>MEM.BW (MB/S)</td>
<td>90.77</td>
<td>182.65</td>
<td>193.32</td>
<td>136.33</td>
<td>236.15</td>
<td>295.71</td>
</tr>
</tbody>
</table>

Code Linking

<table>
<thead>
<tr>
<th>CLUSTER</th>
<th>CODE SECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>solve_nn.m.f[2037 - 2210]</td>
</tr>
<tr>
<td>2</td>
<td>solve_nn.m.f[1478 - 1782]</td>
</tr>
<tr>
<td>3</td>
<td>solve_nn.m.f[1241 - 1345]</td>
</tr>
<tr>
<td>4</td>
<td>solve_nn.m.f[2771 - 2865]</td>
</tr>
<tr>
<td>5</td>
<td>solve_nn.m.f[2388 - 2489]</td>
</tr>
<tr>
<td>6</td>
<td>solve_nn.m.f[1607 - 1633]</td>
</tr>
</tbody>
</table>
Using clusters to understand apps behavior (GROMACS)

Instructions imbalance

IPC Imbalance
Using clusters to understand apps behavior (GROMACS)

64 procs

256 procs
Identifying main code regions (PARSEK)

- duration vs. cluster
- instr. vs. cluster
Outline

- Computation Structure detection
  - Short intro
  - **Aggregative Refinement**
  - Tracking program evolution
  - Scaling clustering algorithm

- Instantaneous performance metric
  - Clustering + Folding
DBSCAN characteristics

Two parameters

- Epsilon: search radius
- MinPoints: minimum cluster density
DBSCAN $Eps$ selection

Which results are better?

$Eps=0.0140$ (Low Value)

$Eps=0.0400$ (High Value)
**DBSCAN single $Eps$ limitation**

**DBSCAN $Eps=0.05$**

- Instructions Completed
- IPC

- Noise
- Cluster 1
- Cluster 2
- Cluster 3
- Cluster 4

**Desired results**

- Instructions Completed
- IPC

- Noise
- Cluster 1
- Cluster 2
- Cluster 3
- Cluster 4
- Cluster 6
Refinement Algorithm Approach

- Analogy between DBSCAN and hierarchical clustering
  - Iterative bottom up construction of a pseudo-dendogram
- Cluster Sequence Score as target
  - Similar to $X$-means approach to decide $K$-means $k$ parameter

Automatic Refinement of Parallel Application Structure Detection (ICPADS 2011)
VAC4 128 tasks
VAC4 128 tasks
VAC4 128 tasks
Outline

- Computation Structure detection
  - Short intro
  - Aggregative Refinement
  - Tracking program evolution
  - Scaling clustering algorithm

- Instantaneous performance metric
  - Clustering + Folding
Correlating multiple runs

- Use and correlate information from different runs
  - Analysis of input parameters
  - Code improvements
  - Using different machines, compilers, flags, libraries
  - Scalability studies
  - Even for the same run: time evolution

- Scatter plot = performance picture
  - Identifies objects and their weight
  - Correlation → image tracking

- Based on heuristics
  - Code regions evolve smoothly (things keep closer)
  - No common callstack means not the same region
  - Time sequence identify regions within and between runs

On the usefulness of object tracking techniques in performance analysis (UPC-DAC-RR-2012-18)
Scenario 1: Analysing scalability (WRF)

128 vs. 256

DBSCAN (Eps=0.018, MinPoints=10) Trace 'WRF.MN.128p.chop2.clustered2.prv'

DBSCAN (Eps=0.018, MinPoints=10) Trace 'WRF.MN.256p.chop2.clustered2.prv'
Scenario 2: Comparing machines & compilers (CG-POP)

**PowerPC, gfortran**

**PowerPC, xlc**

**Intel, gfortran**

**Intel, ifort**
Scenario 3: Problem size impact (NAS-BT)

Class W

Class A

Class C

Class B
Outline

- Computation Structure detection
  - Short intro
  - Aggregative Refinement
  - Tracking program evolution
  - Scaling clustering algorithm

- Instantaneous performance metric
  - Clustering + Folding
First target: online clustering
- Centralised approach (global clustering at the MRNet frontend)
- Data reduction through sampling
- Data classification based on the samples clustering

All processes

32 representatives (50%)

8 representatives + 15% random

25% random records

15% random records

75% less data
6s down from 2m
1. Local clustering
   - Up to 20-30k points per local process
2. Generate models
   - Convex hull, medial axis…
3. Merge the hulls over the MRNet
   - Intersect?
4. Broadcast the global model
5. Classify data locally using the global model
   - Point inside the hull?

PEPC 4K tasks, 3095134 points, 273 tasks (16 way tree, 256 leaves, 12k points per local clustering) → clustering time 28.6 sec
Comparison (parsek)

31515 points

Sampling 25%
Comparison (WRF)

74240 points

Sequential / Par (4+1)  Parallel (2+1)  Sampling 25%

WRF Total time  WRF Clustering time
Computation Structure detection
- Short intro
- Aggregative Refinement
- Tracking program evolution
- Scaling clustering algorithm

Instantaneous performance metric
- Clustering + Folding
Can I get very detailed perf. data with low overhead?

- Application granularity vs. detailed granularity
  - Samples: hardware counters + callstack
- Folding: based on known structure: iterations, routines, clusters;
  - Project all samples into one instance
- Extremely detailed time evolution of hardware counts, rates and callstack with minimal overhead
  - Correlate many counters
  - Instantaneous CPI stack models

Unveiling Internal Evolution of Parallel Application Computation Phases (ICPP 2011)
Correlating with sources: which line should I look?

Folded source code line

Folded instructions
The “benefits” of Fortran 90 intrinsic (PEPC)

Performance metrics

- 96 MIPS
- 2.3 M L2 misses/s
- 0.1 M TLB misses/s

Changes

- 70% time
- 18% instructions
- 63% L2 misses
- 78% TLB misses

253 MIPS (+163%)

do i = 1, n
  htable(i)%node = 0
  htable(i)%key = 0
  htable(i)%link = -1
  htable(i)%leaves = 0
  htable(i)%childcode = 0
End do
Interchanging loops (MR. GENESIS)

Framework for a Productive Performance Optimization
(UPC-DAC-RR-2012-2)
Pre-computing float data – loop split (PMEMD)
Conclusions

Performance analytics
- Data analytics applied to raw performance data
- From data to insight
  - Information is on variability and distribution
- Huge room for research

Showed results of some techniques
- Clustering enables focusing the analysis and open many different uses on the analysis
- Folding makes possible to compute instantaneous performance metric functions with low overhead
- Tracking helps detecting movement in the performance space
  - Sequence of “frames” along many factors (not just time)

www.bsc.es/paraver