Performance Analysis and Tuning of Automatically Parallelized OpenMP Applications

Dheya Mustafa
June 15/2011
Purdue University
Introduction

- Automatic Parallelization
- Offline Empirical Tuning
- Search Space Pruning
Tuning Window

- **Zoom in:** Program Partitioning
  - Most important interactions are between neighboring loops
- **Reduce size of search space dimension**
  - For 100 loops program: $2^{100}$ vs $2^{tuning\ window\ size}$
- **Binary Tuning Option**
  - Loop parallelize: parallelizes small loops
  - Function inline: complicates expressions and makes analysis difficult.
Window based Tuning System Flow

- Serial Program
- Program Partitioning
- Tuning Window Configuration Generation
- Version Generation (CETUS)
- Merge Independent windows configurations
- Compose best version
- Empirical measurement

Tuning Window size = 3

Tuning Options:
- Loop parallelize
- Function inline

Space Navigation Version Generation Empirical Measurement
void main()
{
  #pragma loop name main#0
  #pragma cetus parallel
  for (i=0; i<N; i++)
  {
  }
  #pragma loop name main#1
  #pragma cetus parallel
  for (i=0; i<N; i++)
  {
  }
  #pragma loop name main#2
  #pragma cetus parallel
  for (i=0; i<N; i++)
  {
  }
  #pragma loop name main#3
  #pragma cetus parallel
  for (i=0; i<N; i++)
  {
  }
}

#pragma loop name main#0
#pragma cetus parallel
for (i=0; i<N; i++){}
#pragma loop name main#1
for (i=0; i<N; i++){}
#pragma loop name main#2
#pragma ncinlinein
#pragma cetus parallel
for (i=0; i<N; i++){}
#pragma loop name main#3
#pragma cetus parallel
for (i=0; i<N; i++){}
#pragma loop name main#3
#pragma cetus parallel
for (i=0; i<N; i++){}

#endif CETUS_TIMING
cetus_toc(&cetus_prof, 2);
#endif
#pragma loop name main#3
#pragma cetus parallel
for (i=0; i<N; i++){}
#pragma CETUS_TIMING
cetus_toc(&cetus_prof, 2);
#pragma loop name main#3
#pragma cetus parallel
for (i=0; i<N; i++){}
Parallel Coverage

- Parallel Coverage exhibits the potential parallel performance.
- Benchmarks with low parallel coverage are not amenable to automatic parallelization.
- LU benchmark.
- Gives an intuition about representativeness of dataset.
Performance Evaluation

![Graphs showing speedup comparisons for CG, SP, BT, and EP categories with different parallelization methods: Serial, All Parallel, Related Work, Window Tuned, and Hand Parallel.](image)

Legend:
- Serial
- All Parallel
- Related Work (Reference)
- Window Tuned
- Hand Parallel
Performance Evaluation (II)

LU

MG

IS

FT

Serial  All Parallel  Related Work (Reference)  Window Tuned  Hand Parallel
Performance Evaluation (III)

- Unexpected performance degradation in ART
- Parallel coverage using training dataset is high
## Tuning Time

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Related Work</th>
<th>Window tuned</th>
<th>Number of loops</th>
</tr>
</thead>
<tbody>
<tr>
<td>BT</td>
<td>255</td>
<td>$32 = 2^{2+3} = 2^5$</td>
<td>65</td>
</tr>
<tr>
<td>CG</td>
<td>26</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>EP</td>
<td>10</td>
<td>32</td>
<td>6</td>
</tr>
<tr>
<td>FT</td>
<td>27</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>LU</td>
<td>173</td>
<td>32</td>
<td>43</td>
</tr>
<tr>
<td>MG</td>
<td>39</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>SP</td>
<td>558</td>
<td>32</td>
<td>102</td>
</tr>
<tr>
<td>EQUAKE</td>
<td>21</td>
<td>32</td>
<td>9</td>
</tr>
<tr>
<td>ART</td>
<td>79</td>
<td>32</td>
<td>29</td>
</tr>
</tbody>
</table>
Conclusion

- We have presented an automatic tuning framework with pruned search space that outperforms state of the art reference point in performance as well as tuning time.
- Our tuning framework partitions a program into sections and tunes each section individually.
- Promising preliminary results open the door to window based tuning.
Questions