Intra-Application Data-Communication Characterization

Imran Ashraf,
Vlad Mihai Sima,
Koen Bertels

Computer Engineering Lab,
TU Delft, The Netherlands
Trends

- Growing demand of processing
- Growing number of transistors per chip
- Increasing the clock rate not feasible
  - Fabrication cost
  - Power consumption
- Trend is increasing number of homogeneous and heterogeneous cores
Multicore Processor Proliferation

- Intel Haswell E5 (18)
- AMD Opteron Abu Dhabi (16)
- ARM Cortex A57 (4)
- ORACLE SPARCT5 (16)
- IBM Power 8 (12, 2013)
- IBM Power 7 (8)
- IBM Power 4 (2, 2001)
- Intel Xeon Hapertown (4)
- Sun UltraSPARC T2 (8)
- Intel Core i7 Gulftown (6)
- AMD Opteron Interlagos (12)

Timeline:
- 2000
- 2005
- 2010
- 2015

# Cores
- Intel
- AMD
- IBM
- SONY
- ARM

50 Cores
Multicore Processor Proliferation

- Embedded Systems
- Servers & Workstations
- Personal Computer
- Supercomputers
Application Partitioning

Application
- Part 0
- Part 1
- Part 2
- Part 3

Architecture
- Core 0
- Core 1
- Core 2
- Core 3
Cost of Communication and Memory-Assignment

Communication and improper memory-assignment may reduce anticipated performance improvement.

Tools are required to provide detailed communication profile and highlight memory access patterns to perform memory assignment.

GPU Memory Hierarchy
- Constant Memory
- Shared Memory
- Texture Memory
- Global Memory

Intra-Application Data-Communication Characterization
ExaComm 2015, Co-located with ISC 2015, Frankfurt, Germany
Existing Profilers

- Profilers focusing on computational hot-spots, e.g. gprof, oprofile, callgrind, zoom, intel vtune ...
- Memory Profilers, e.g. cachegrind, oprofile, Intel Vtune, AMD codeXL
- Communication Profilers
  - Architecture dependent, CETA
  - Mostly for existing parallel applications, e.g. Vampire, TAU, HPCToolkit ...
  - Very high space/time overhead, e.g. Quad and Pincomm

MCProf : Run-time Communication Profiler, Architecture Independent, order of magnitude less overhead
MCPROF: Memory and Communication Profiler

- Run-time open-source profiler based on Intel Pin framework
- Traces memory reads/writes to report
  - Compute/Memory intensive functions and objects
  - Data communication at function/loop granularity
- The output in various formats:
  - Flat profile
  - Communication Matrix
  - Communication Graph

https://bitbucket.org/imranashraf/mcprof
MCProf Example

```c
#define SIZE 100
int *srcArr1, *srcArr2,
    *sumArr, *diffArr;
void initVecs() {
    for (i = 0; i < SIZE; i++) {
        *(srcArr1+i)=i*5 + 7;
        *(srcArr2+i)=2*i - 3;
    }
}
int main() {
    srcArr1 = malloc( SIZE*sizeof(int) );
    //similarly, other allocations
    initVecs();
    for(j=0;j<3;j++) sumVecs();
    for(j=0;j<5;j++) diffVecs();
    printf("%d",sumArr[1]+diffArr[1]);
    free(srcArr1);
    // similarly, other memory frees
    return 0;
}
```
MCProf Example

- Ovals represent functions
  - Name
  - Dynamically executed Instructions
  - No. of calls
- Rectangles represent objects
  - Name
  - Size
- Edges Represent communication
MCProf: Block Diagram

Memory Access Tracer
- Instruction-Level Instrumentation
- Image-Level Instrumentation
- Routine-Level Instrumentation

Data Collection Engines
- Engine 1
- Engine 2
- ... Engine N

Write Addr, Write Size, Producer
Read Addr, Producer

Shadow Memory Unit
- Table Lookup
- Direct Mapping

Collected Information
- Inter Func. Comm. Matrix
- Inter Thread Comm. Matrix
- ...

called Information

Controller

Communication Profile
- DOT
- Comm. Matrix
- XML

Pin
- Instrumentation APIs

VM
- JIT Compiler
- Emulation Unit

Call-stack
- Symbol Table

Pin Instrumentation APIs

VM JIT Compiler Emulation Unit

Intra-Application Data-Communication Characterization
ExaComm 2015, Co-located with ISC 2015, Frankfurt, Germany
MCProf: Basic Idea

Challenge: Read/Write can happen anywhere in 128TB address space

Efficient shadow memory is critical for the tool's performance
MCProf: Hybrid Shadow Memory

\[
\text{Shadow Addr} = ((\text{Addr} \land M0H) \ll \log_2 (\text{SCALE})) + (\text{Addr} \land (SM1L + SM0L)) + SM0L
\]
Case-study: KLT Feature Tracker

- Kanade-Lucas-Tomasi (KLT) Feature Tracking
- Version 1.3.4 (latest version)
  - 102 functions
  - 17 source-files
  - 5033 lines of code

- Mapping KLT Application to GPU
- Focus is on utilizing the MCPROF output:
  - To optimize communication
  - Better memory assignment
Experimental Setup

- 2.5 GHz Intel(R) Xeon(R) CPU with 32 GB RAM
- Nvidia GeForce GT 640 GPU with 2 GB memory
- Ubuntu 12.04 is running on the machine with Linux kernel 2.6.32-24-server
- Nvidia driver version 319.37
- Nvidia CUDA toolkit V 6.0
# KLT Kernels

<table>
<thead>
<tr>
<th>Function Name</th>
<th>%Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLTSelectGoodFeatures</td>
<td>54.07</td>
</tr>
<tr>
<td>convolveImageVert</td>
<td>19.65</td>
</tr>
<tr>
<td>convolveImageHoriz</td>
<td>10.17</td>
</tr>
<tr>
<td>trackfeature</td>
<td>7.81</td>
</tr>
<tr>
<td>%Total Contribution</td>
<td>91.7</td>
</tr>
</tbody>
</table>

 Theoretical Speedup ~12x(336,860,677,890)
## Speedup: w/o Communication Optimization

<table>
<thead>
<tr>
<th>Kernel</th>
<th>CPU Time (Sec)</th>
<th>GPU Time (Sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GPU Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kernel</td>
<td>Memory</td>
</tr>
<tr>
<td>SelectFeatures</td>
<td>0.452974</td>
<td>0.038965</td>
<td>0.012727</td>
</tr>
<tr>
<td>TrackFeatures</td>
<td>0.104875</td>
<td>0.053329</td>
<td>0.017402</td>
</tr>
<tr>
<td>ConvolveHoriz</td>
<td>0.010136</td>
<td>0.000023</td>
<td>0.005315</td>
</tr>
<tr>
<td>ConvolveVert</td>
<td>0.021814</td>
<td>0.000022</td>
<td>0.005315</td>
</tr>
</tbody>
</table>

High kernel speedup but very low total speedup due to slow communication
trackfeature should be on GPU to reduce the bulk of communication

KLTToFloatImage is not kernel but should be mapped to GPU
# Hot Objects and Memory Assignment

<table>
<thead>
<tr>
<th>Objects</th>
<th>Reads</th>
<th>Writes</th>
<th>Total</th>
<th>Percent</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmpimgCS</td>
<td>3.86E+08</td>
<td>5.19E+07</td>
<td>4.38E+08</td>
<td>26.6</td>
<td>7.4</td>
</tr>
<tr>
<td>pointlist</td>
<td>1.34E+08</td>
<td>1.34E+08</td>
<td>2.68E+08</td>
<td>16.3</td>
<td>1</td>
</tr>
<tr>
<td>pyramidImg</td>
<td>1.34E+08</td>
<td>3.54E+07</td>
<td>1.70E+08</td>
<td>10.3</td>
<td>3.8</td>
</tr>
<tr>
<td>grady</td>
<td>1.34E+08</td>
<td>3.15E+06</td>
<td>1.37E+08</td>
<td>8.3</td>
<td>42.7</td>
</tr>
<tr>
<td>gradx</td>
<td>1.34E+08</td>
<td>3.15E+06</td>
<td>1.37E+08</td>
<td>8.3</td>
<td>42.7</td>
</tr>
<tr>
<td>tmpimgTF</td>
<td>6.71E+07</td>
<td>9.44E+06</td>
<td>7.65E+07</td>
<td>4.6</td>
<td>7.1</td>
</tr>
<tr>
<td>gaussderiv_kernel</td>
<td>6.71E+07</td>
<td>4.77E+03</td>
<td>6.71E+07</td>
<td>4.1</td>
<td>14063.1</td>
</tr>
<tr>
<td>gauss_kernel</td>
<td>6.71E+07</td>
<td>4.63E+03</td>
<td>6.71E+07</td>
<td>4.1</td>
<td>14500.6</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>82.6</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Map to Shared Memory**
- **Keep in Device Memory**
- **Map to Constant Memory**
Achieved Speedup

![Achieved Speedup Diagram](image)

Intra-Application Data-Communication Characterization
ExaComm 2015, Co-located with ISC 2015, Frankfurt, Germany
Execution-time Overhead Comparison

![Bar chart comparing execution-time overhead for different applications. The applications plotted are canny, KLT, ocean-NC, fmm, raytrace, bwa-mem, and Mean. The chart shows overhead in log scale, with the overhead values ranging from 1 to 10,000. The overhead is compared across four different tools: Pincomm, QUAD, MCPROF, and MCPROF\textsubscript{x}. The values are represented by bars in blue, yellow, green, and red colors respectively. The overhead values are indicated at the top of the bars: 3530x for canny, 1855x for KLT, 34x for ocean-NC, 83x for fmm, 83x for raytrace, 54x for bwa-mem, and the mean of 83x.](image-url)
Intra-Application Data-Communication Characterization

ExaComm 2015, Co-located with ISC 2015, Frankfurt, Germany

Memory-usage Overhead Comparison

Applications

canny  KLT  ocean-NC  fmm  raytrace  bwa-mem  Mean

Pincomm  QUAD  MCPROF  MCPROFx

Meaning of data: Pincomm, QUAD, MCPROF, MCPROFx

Memory usage overhead (log scale)
Execution-time Overhead Comparison: Data-structure only

![Bar chart comparing execution time for different image sizes with Trie, Hash map, and Hybrid data structures.](chart.png)
Memory-usage Overhead Comparison: Data-structure only

- **Trie**
- **Hash map**
- **Hybrid**

Image Size:
- 120x80
- 320x240
- 640x480
- 1024x768
Conclusion

• MCProf: an open-source runtime communication profiler
  • Architecture independent
  • Manageable space/time overhead

• The provided information was utilized to perform memory assignment and communication aware mapping of KLT on an accelerator based platform

• Communication-aware interconnect design for Hybrid-core computer by Micron

• Communication-aware porting of bio-informatic application to PGAS programming model
Questions

MCPROF: https://bitbucket.org/imranashraf/mcprof