Intra-Application Data-Communication Characterization

Imran Ashraf,

Vlad Mihai Sima,

Koen Bertels

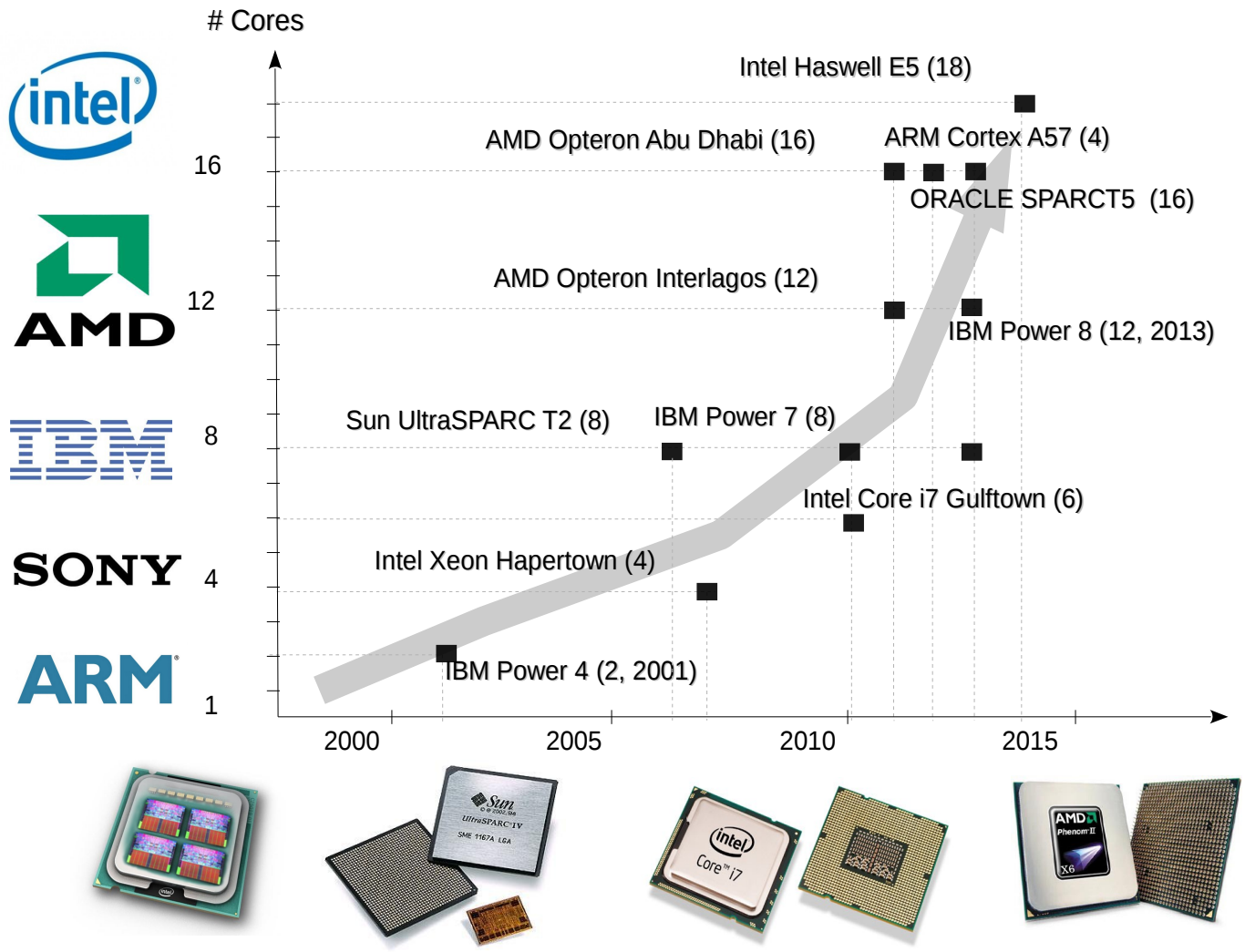
Computer Engineering Lab,

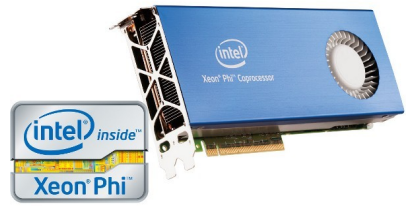
TU Delft, The Netherlands

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

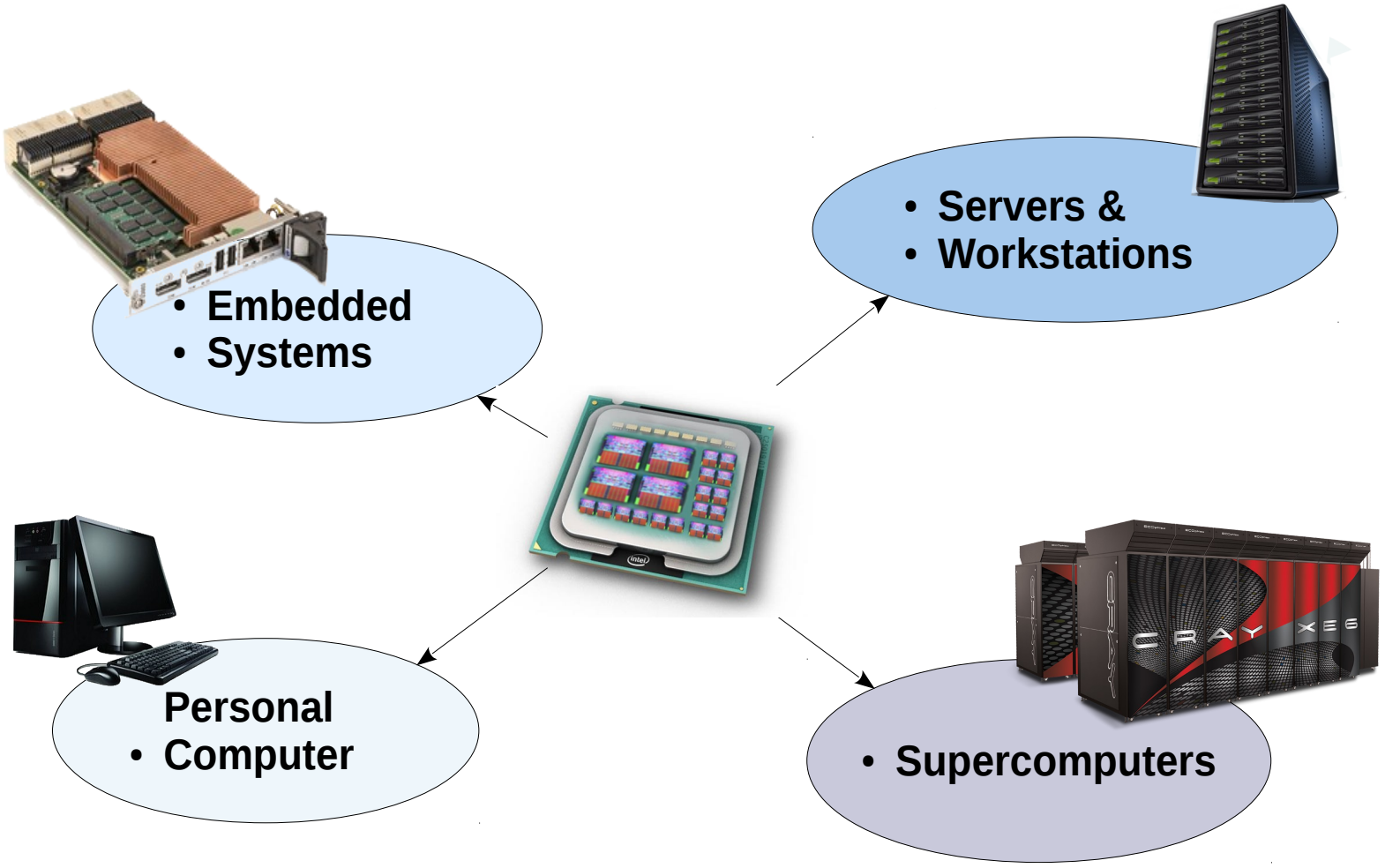
Multicore Processor Proliferation



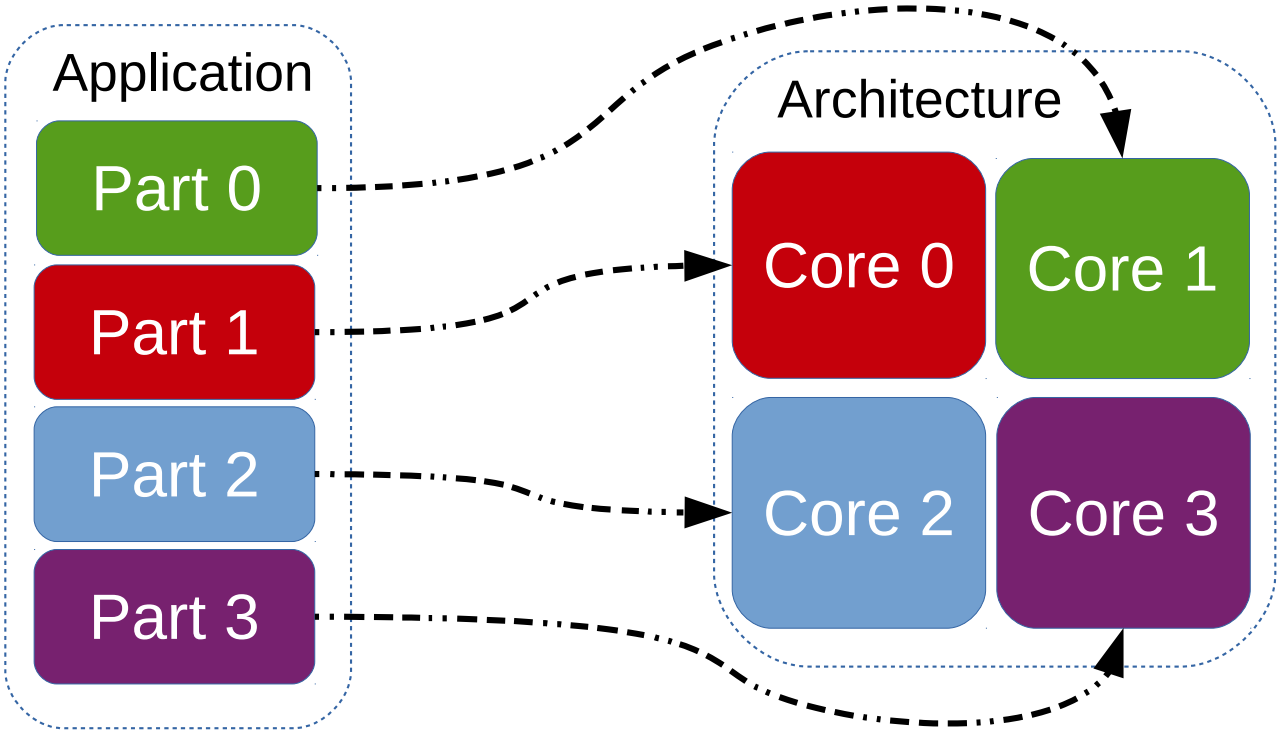


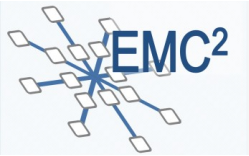
50 Cores

Multicore Processor Proliferation

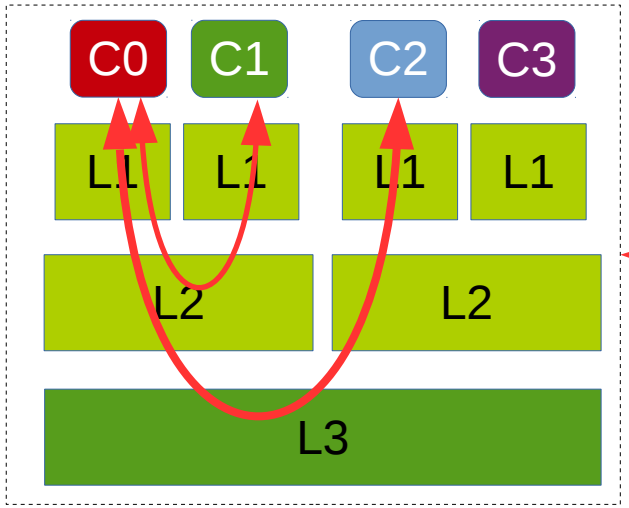


Application Partitioning





Cost of Communication and Memory-Assignment



GPU Memory Hierarchy

- Constant Memory
- Shared Memory
- Texture Memory
- Global Memory

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

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

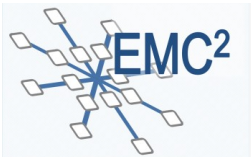
- Run-time open-source profiler based on Intel Pin framework
- Traces memory reads/writes to report
  - Compute/Memory intensive functions and objects
  - Data communication at function/loop granularity
- The output in various formats:
  - Flat profile
  - Communication Matrix
  - Communication Graph

https://bitbucket.org/imranashraf/mcprof

MCProf Example

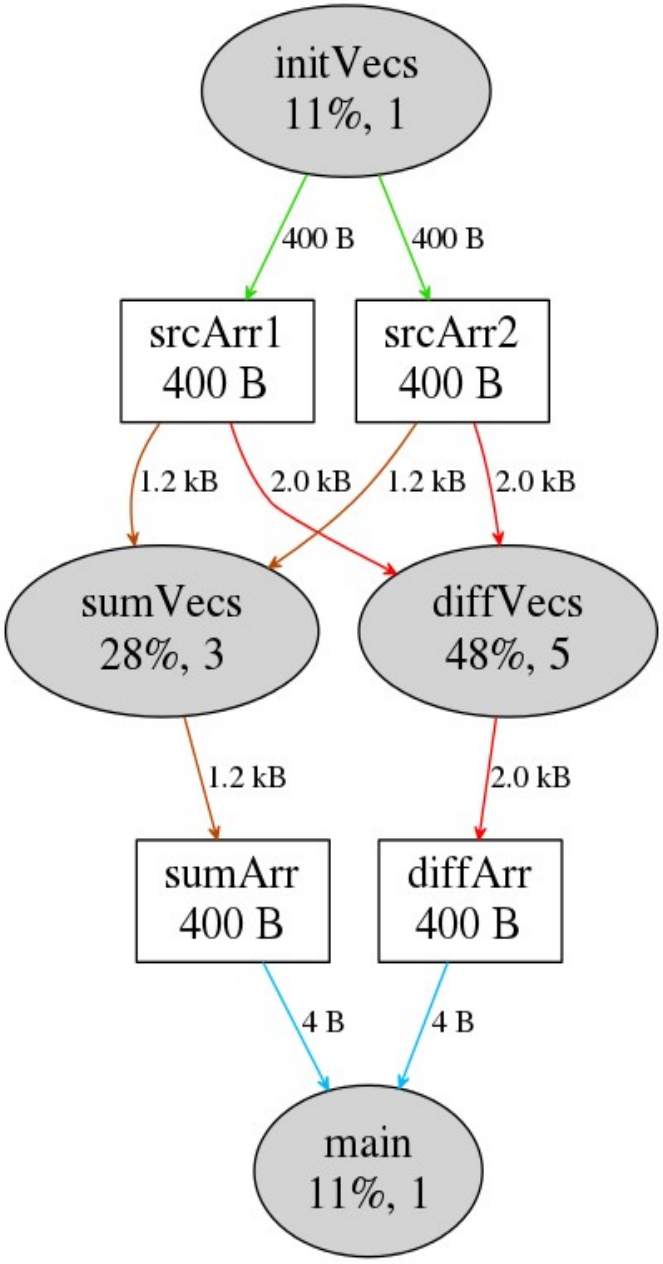
```
#define SIZE 100
int *srcArr1, *srcArr2,
    *sumArr, *diffArr;
void initVecs() {
  for(i = 0; i < SIZE; i++)
  {
    *(srcArr1+i)=i*5 + 7;
    *(srcArr2+i)=2*i - 3;
  }
}
```

```
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;
}
```

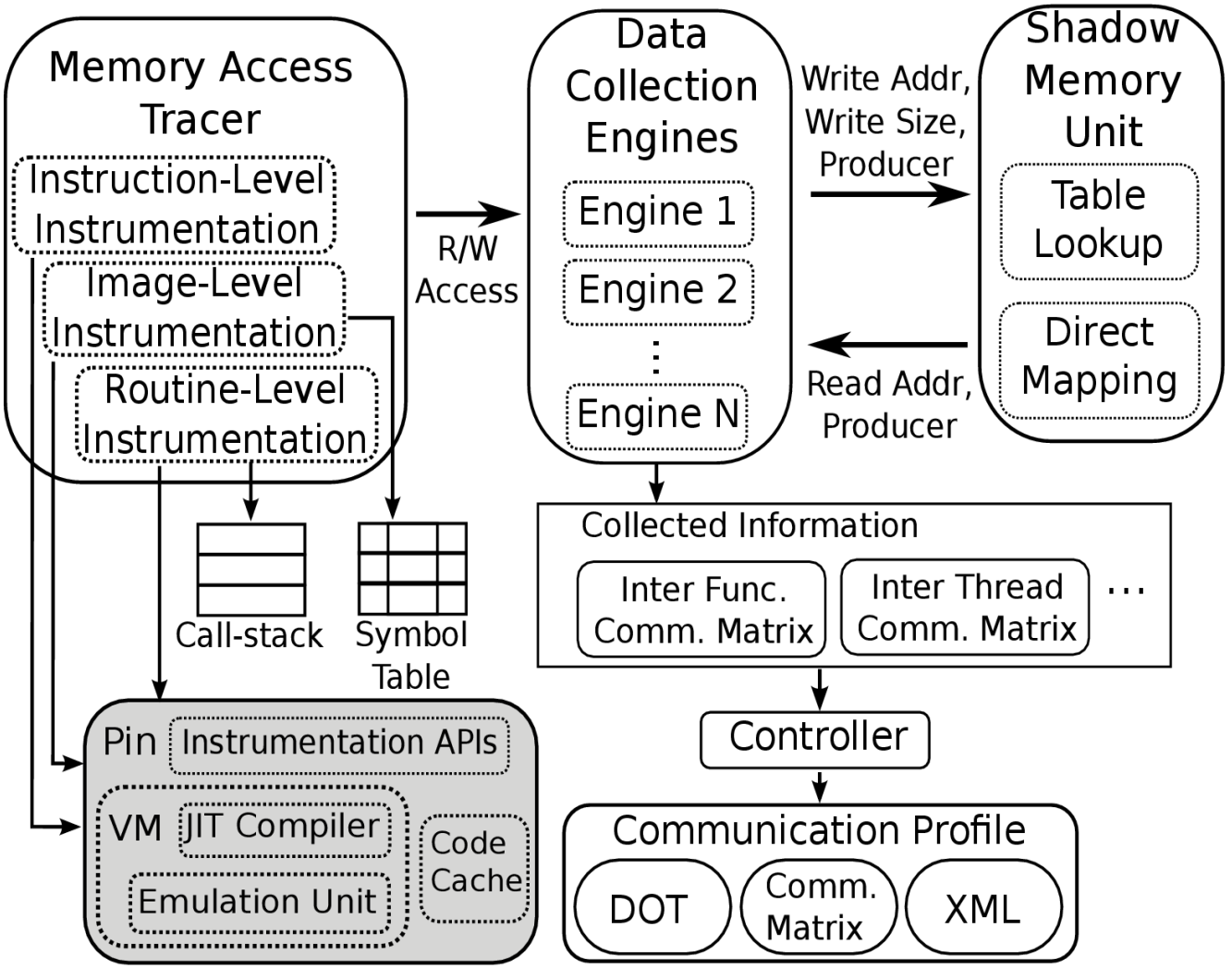



MCProf Example

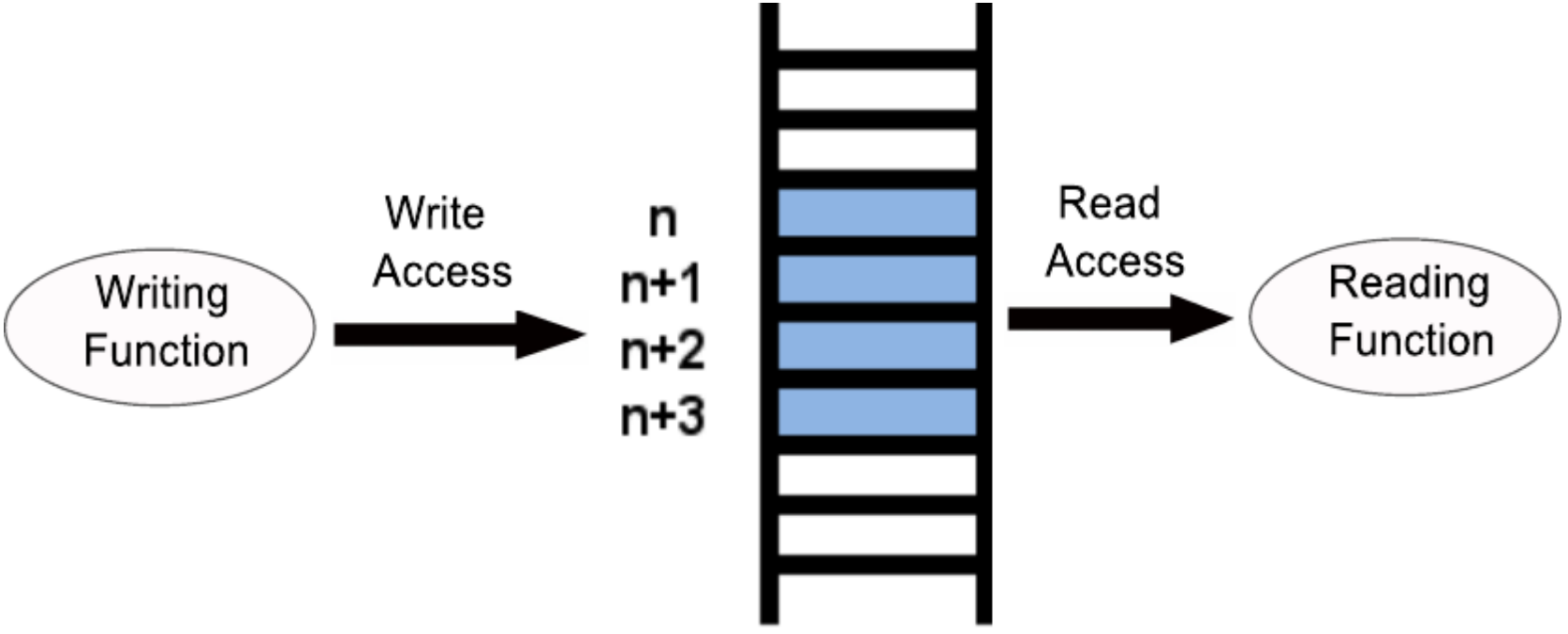
- Ovals represent functions
  - Name
  - Dynamically executed Instructions
  - No. of calls
- Rectangles represent objects
  - Name
  - Size
- Edges Represent communication



MCProf: Block Diagram



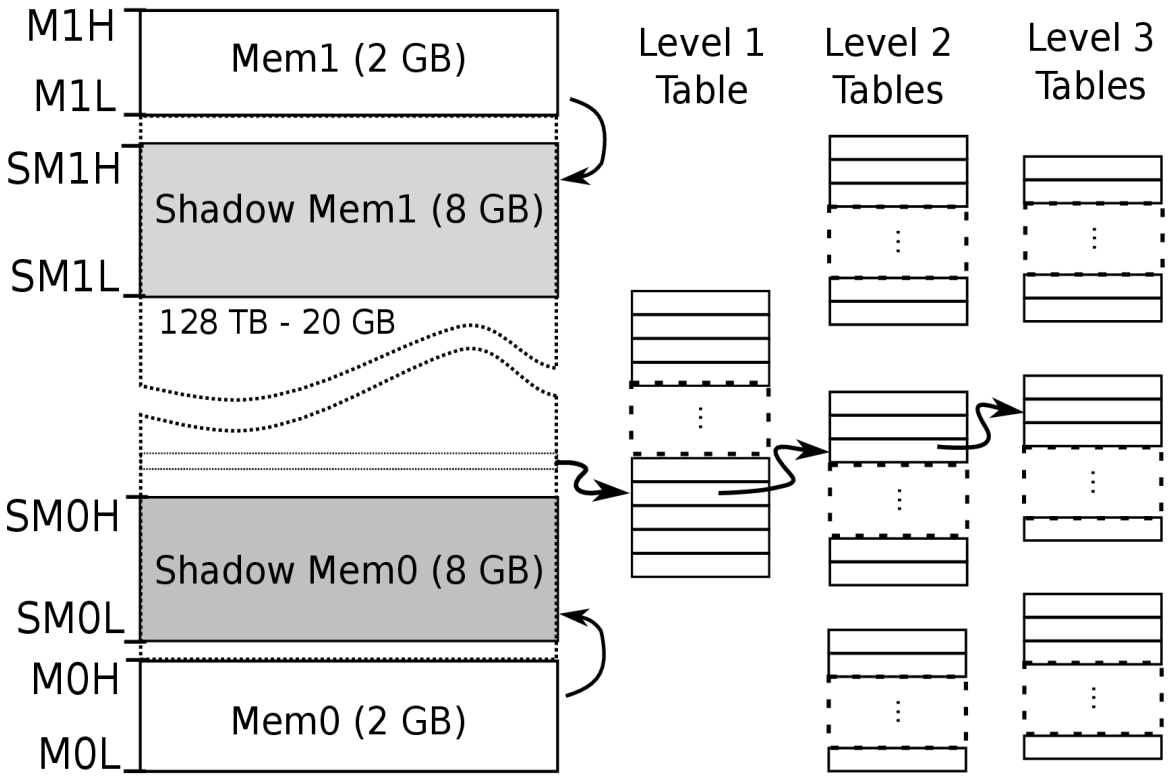
MCProf: Basic Idea



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

Efficient shadow memory is critical for the tool's performance

MCProf: Hybrid Shadow Memory



$$Shadow\ Addr = ((Addr \wedge M0H) \ll \log 2(SCALE)) + (Addr \wedge (SM1L + SM0L)) + SM0L$$

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

Function Name	%Time
KLTSelectGoodFeatures	54.07
convolveImageVert	19.65
convolveImageHoriz	10.17
trackfeature	7.81
%Total Contribution	**91.7**

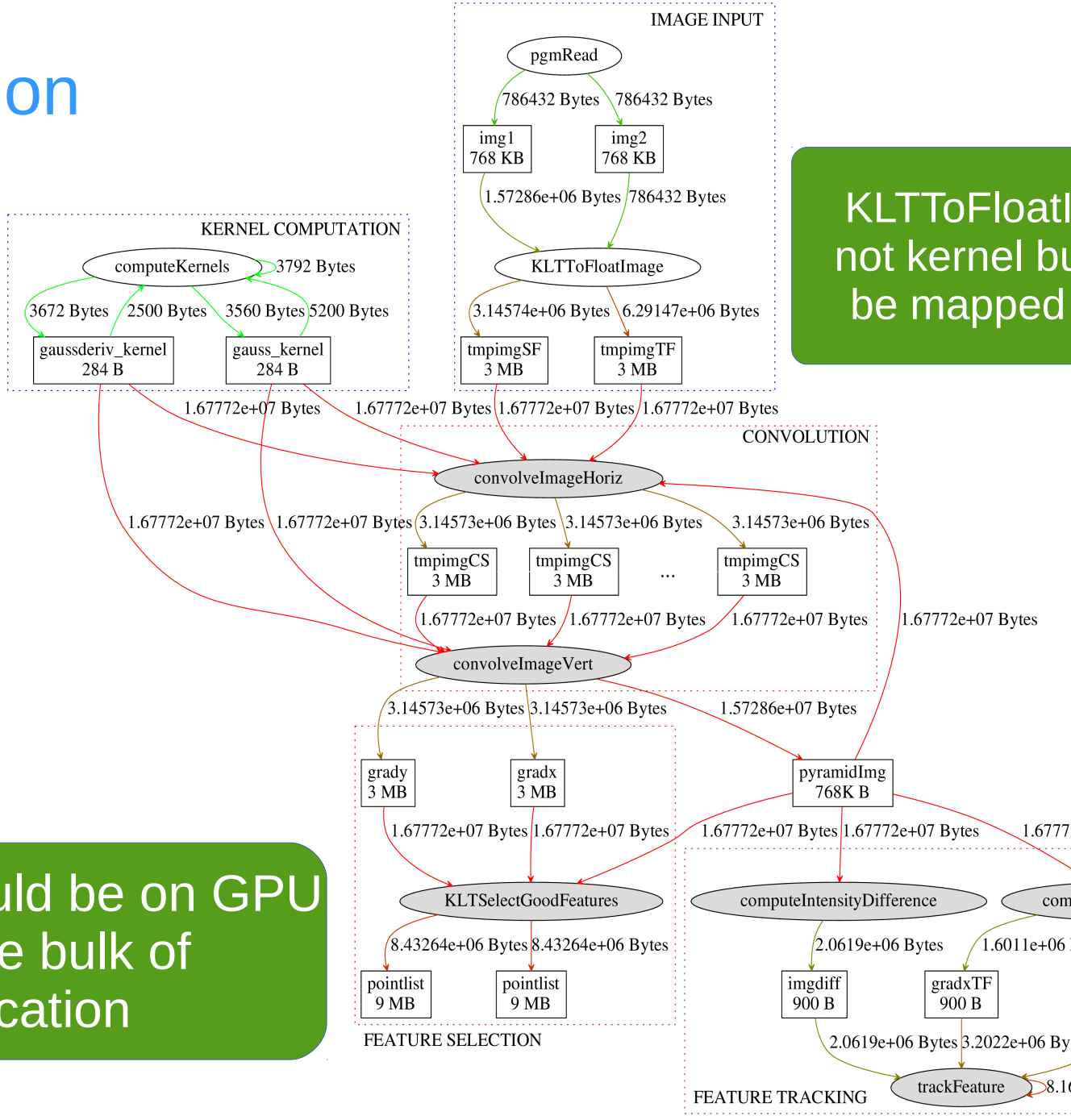
Theoretical Speedup ~12x

Speedup: w/o Communication Optimization

Kernel	CPU Time (Sec)	GPU Time (Sec)		Speedup	
		Kernel	Memory	Kernel	Application
SelectFeatures	0.452974	0.038965	0.012727	11.625150	8.762942
TrackFeatures	0.104875	0.053329	0.017402	1.966566	1.482730
ConvolveHoriz	0.010136	0.000023	0.005315	440.695652	1.898838
ConvolveVert	0.021814	0.000022	0.005315	1090.700	4.088847

High kernel speedup but very low total speedup due to slow communication

Communication Graph



KLTToFloatImage is not kernel but should be mapped to GPU

trackfeature should be on GPU to reduce the bulk of communication

Hot Objects and Memory Assignment

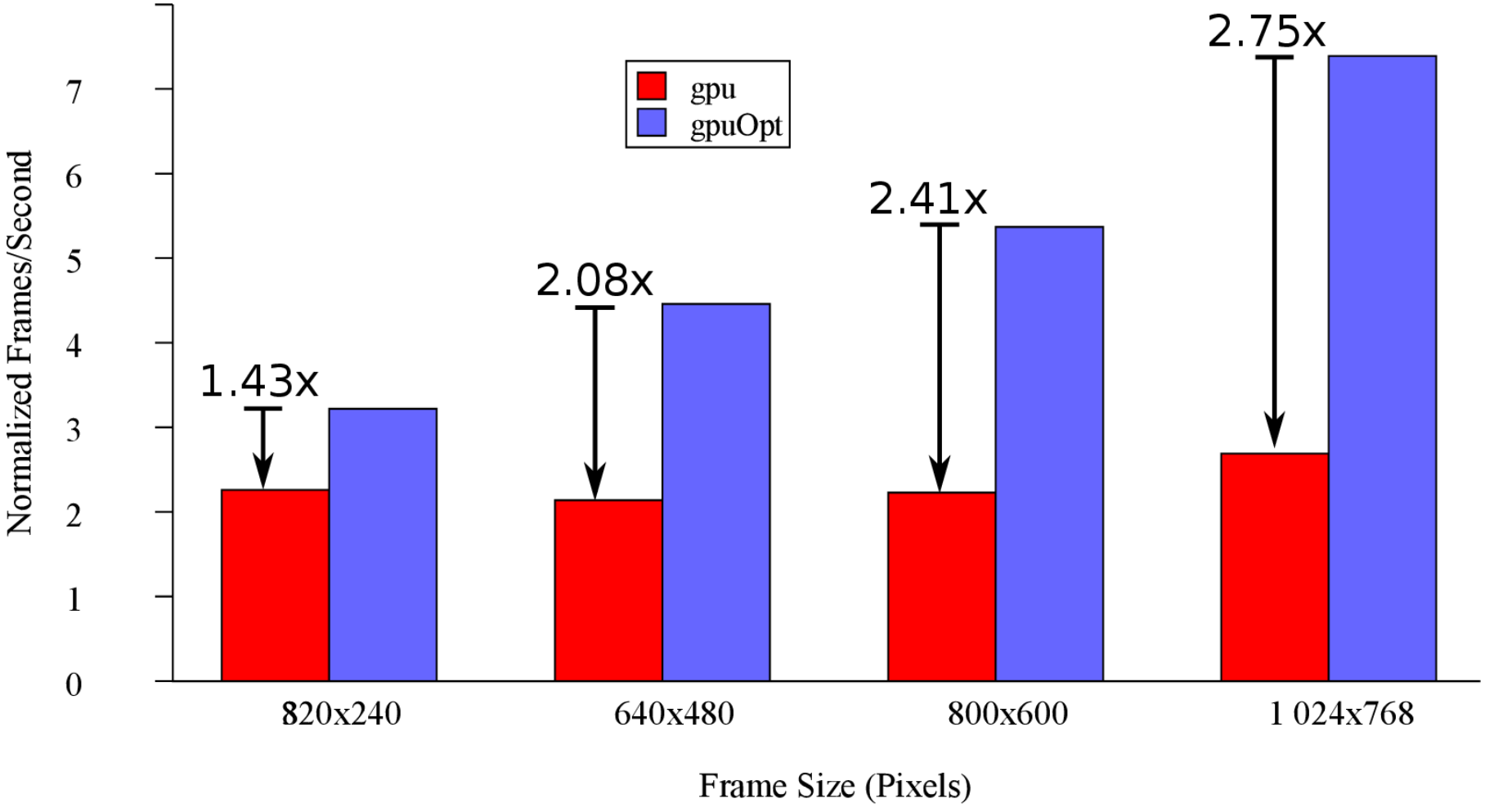
Objects	Reads	Writes	Total	Percent	Read/Write
tmpimgCS	3.86E+08	5.19E+07	4.38E+08	26.6	7.4
pointlist	1.34E+08	1.34E+08	2.68E+08	16.3	1
pyramidImg	1.34E+08	3.54E+07	1.70E+08	10.3	3.8
grady	1.34E+08	3.15E+06	1.37E+08	8.3	42.7
gradx	1.34E+08	3.15E+06	1.37E+08	8.3	42.7
tmpimgTF	6.71E+07	9.44E+06	7.65E+07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

Map to Shared Memory

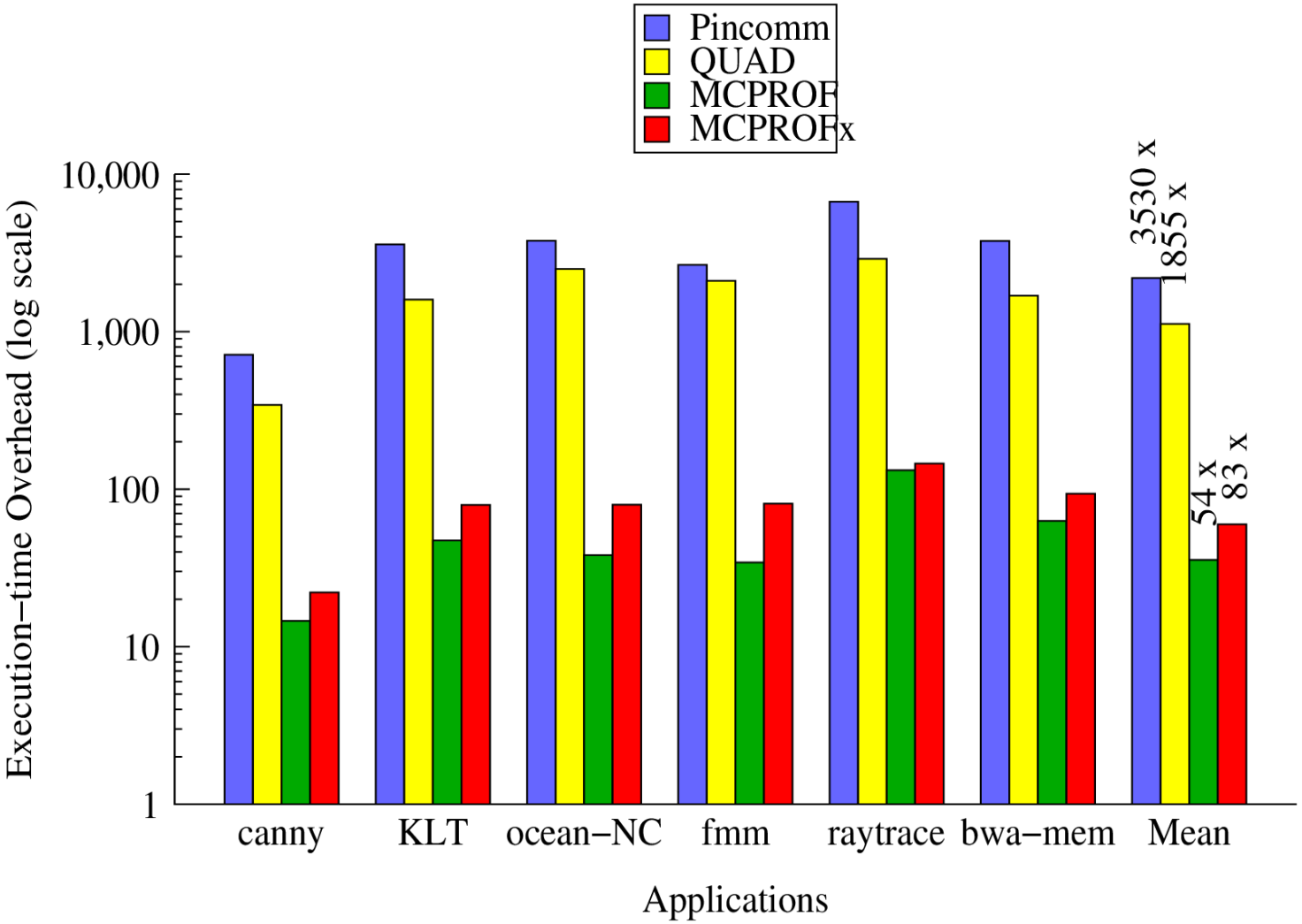
Keep in Device Memory

Map to Constant Memory

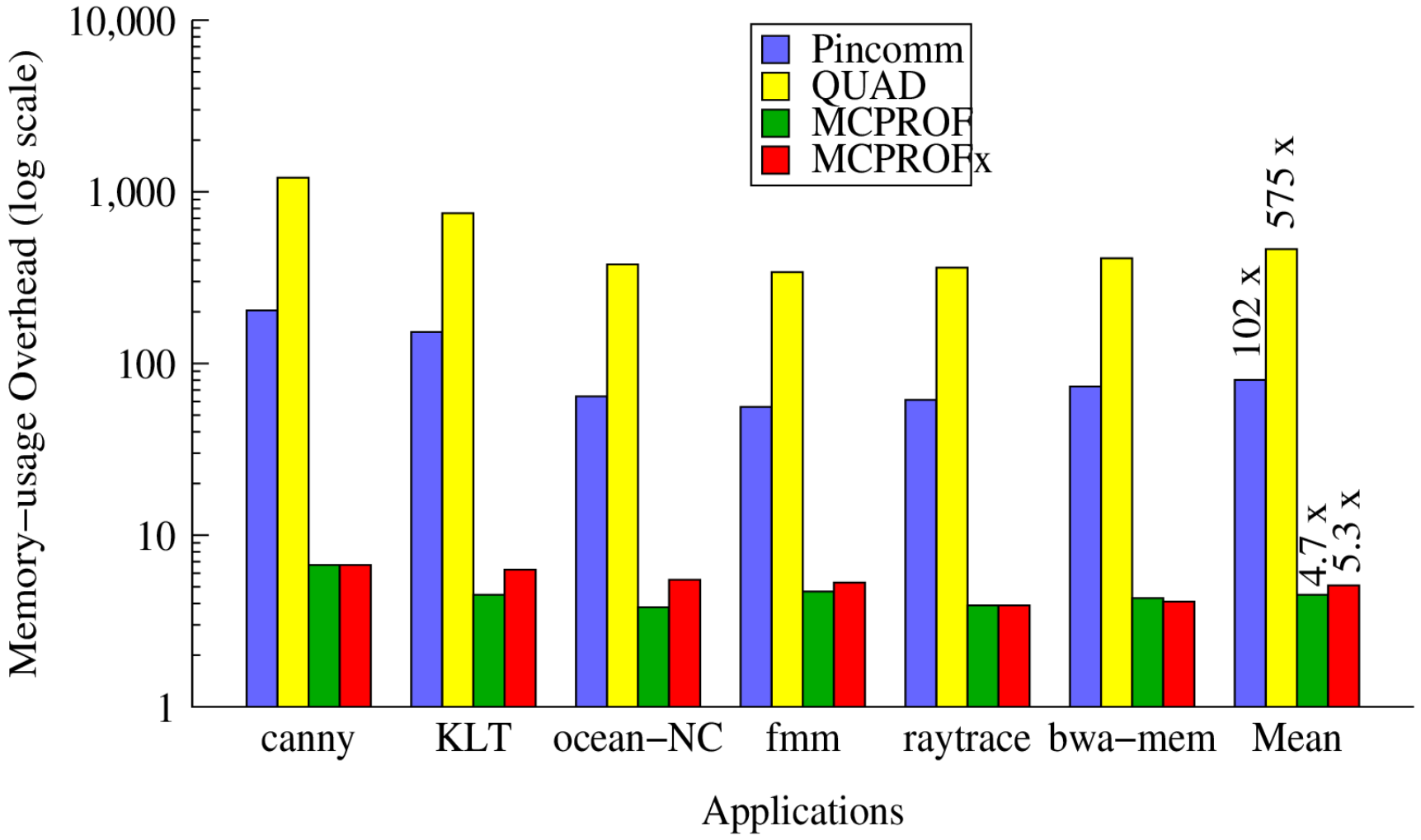
Achieved Speedup



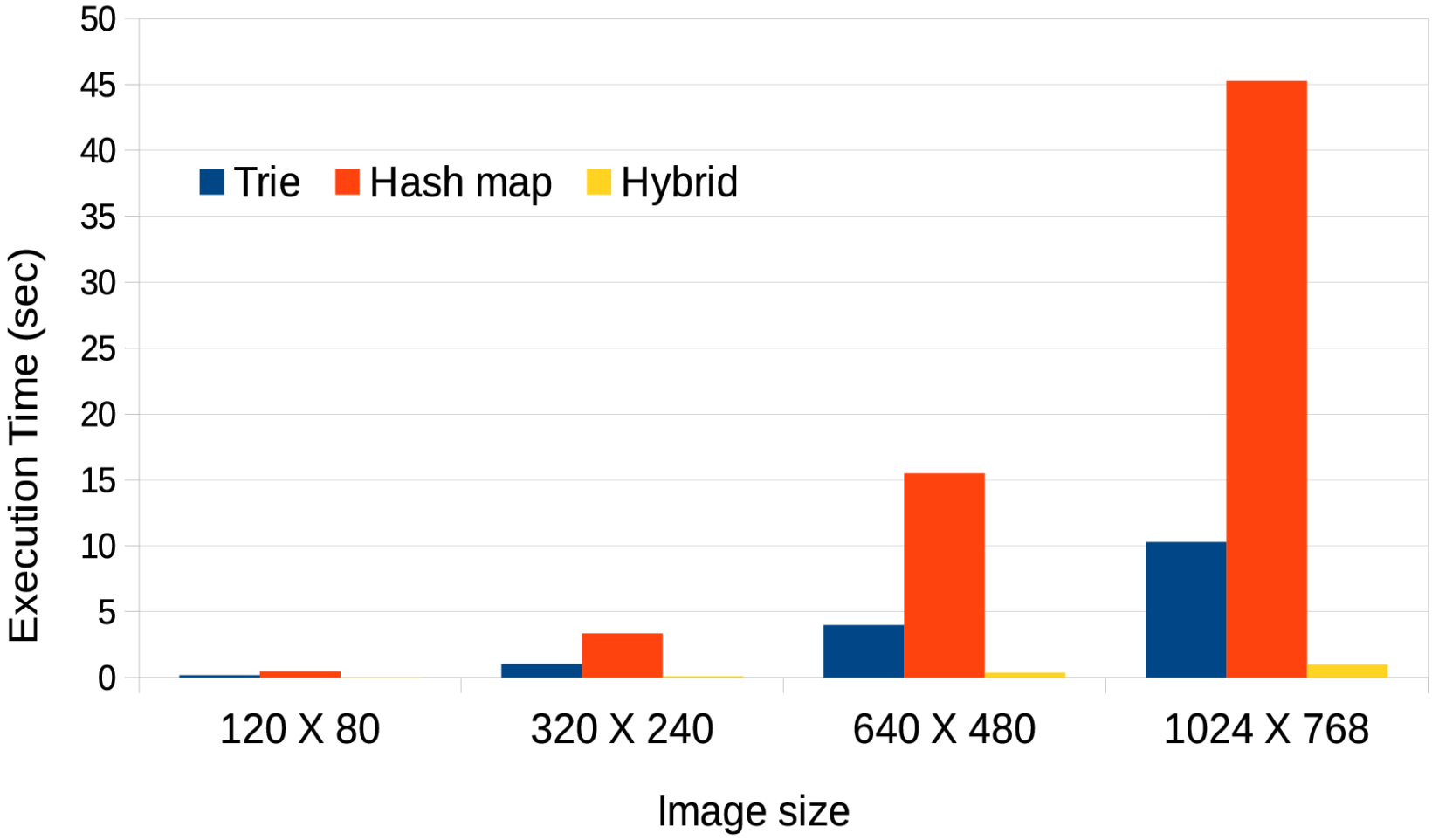
Execution-time Overhead Comparison



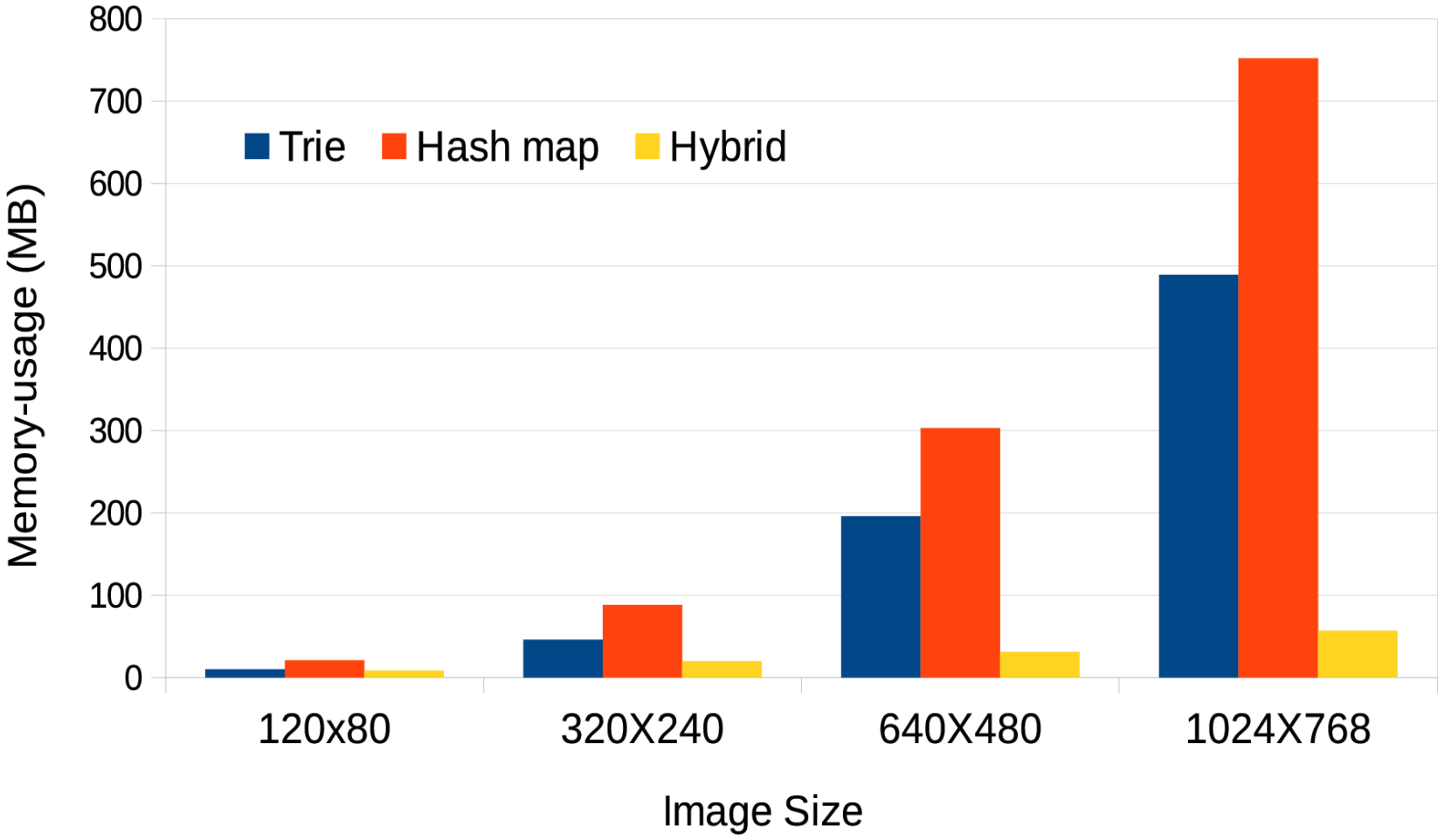
Memory-usage Overhead Comparison



Execution-time Overhead Comparison: Data-structure only



Memory-usage Overhead Comparison: Data-structure only



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*