PAKCK: Power Analysis of Key Computational Kernels

Albert Reuther
Computing and Analytics Group

Suite of Embedded Applications and Kernels (SEAK) Workshop at DAC 2014

1 June 2014

DARPA MTO
PM: Dr. Joseph Cross

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

This work is sponsored by the Defense Advanced Research Projects Agency under Air Force Contract FA8721-05-C-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

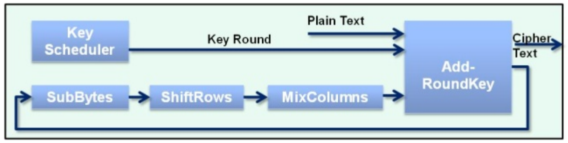
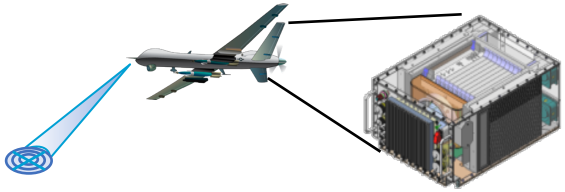
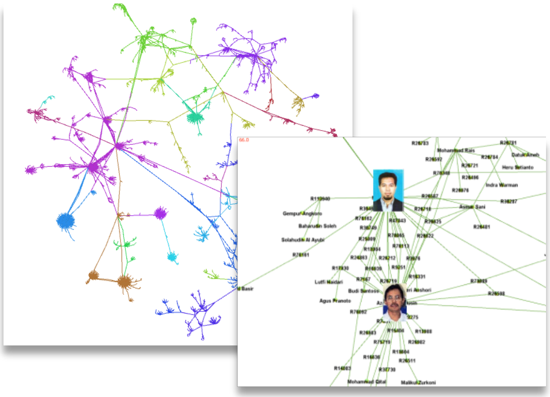
Outline

- Introduction
- Key Computational Kernels and Computational Architectures
- Results
- Exploration of Possible using LLMORE Simulator
- Other Key Benchmark Suites
- Summary and Future Work

PAKCK Overview

- Set of DoD applications surveyed
- Set of key kernels identified/implemented
  - Dense kernels
  - Sparse kernels
- Specific target architectures chosen
  - ASIC
  - FPGA
  - Multicore: CPU and GPU
- Methodologies for power/performance characterization on architectures identified
- Initial power/performance characterization for some kernels/ applications
- Simulation framework LLMORE extended
  - Support for dynamic power models and additional architectures
  - Methodology for power simulations defined
  - Initial experiments of “possible”

Kernel Selection: Three Key DoD Domains

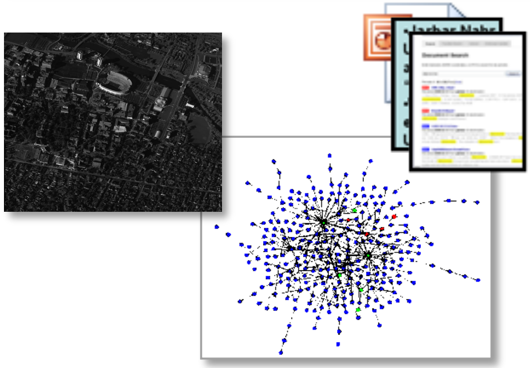
Encryption	Signal & Image Processing	Databases, Big Data, Graph Analytics
  Key Scheduler Key Round Plain Text Cipher Text Add-RoundKey SubBytes ShiftRows MixColumns		
Key kernel: AES	Key kernels: GEMV, FFT, matrix element-wise multiply	Key kernels: SpGEMM, SpGEMV, BFS Additional info

- Surveyed three application domains – key power kernels identified
- Implementations of key kernels gathered/written

GEMV = dense matrix-vector multiplication, SpGEMM = sparse matrix-matrix multiplication, SpGEMV= sparse matrix-vector multiplication, BFS = breadth first search

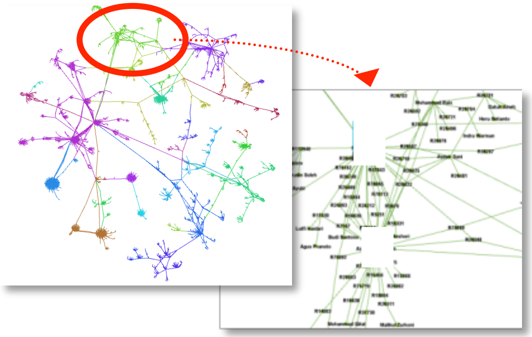
Databases, Big Data, Graph Analysis

ISR



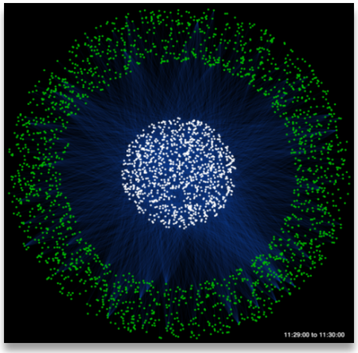
- Graphs represent entities and relationships detected through multi-INT sources
- 1,000s – 1,000,000s tracks and locations
- GOAL: Identify anomalous patterns of life

Social



- Graphs represent relationships between individuals or documents
- 10,000s – 10,000,000s individual and interactions
- GOAL: Identify hidden social networks

Cyber



- Graphs represent communication patterns of computers on a network
- 1,000,000s – 1,000,000,000s network events
- GOAL: Detect cyber attacks or malicious software

Cross-Mission Challenge:
Detection of subtle patterns in massive multi-source noisy datasets

Source: Ben Miller, MITLL

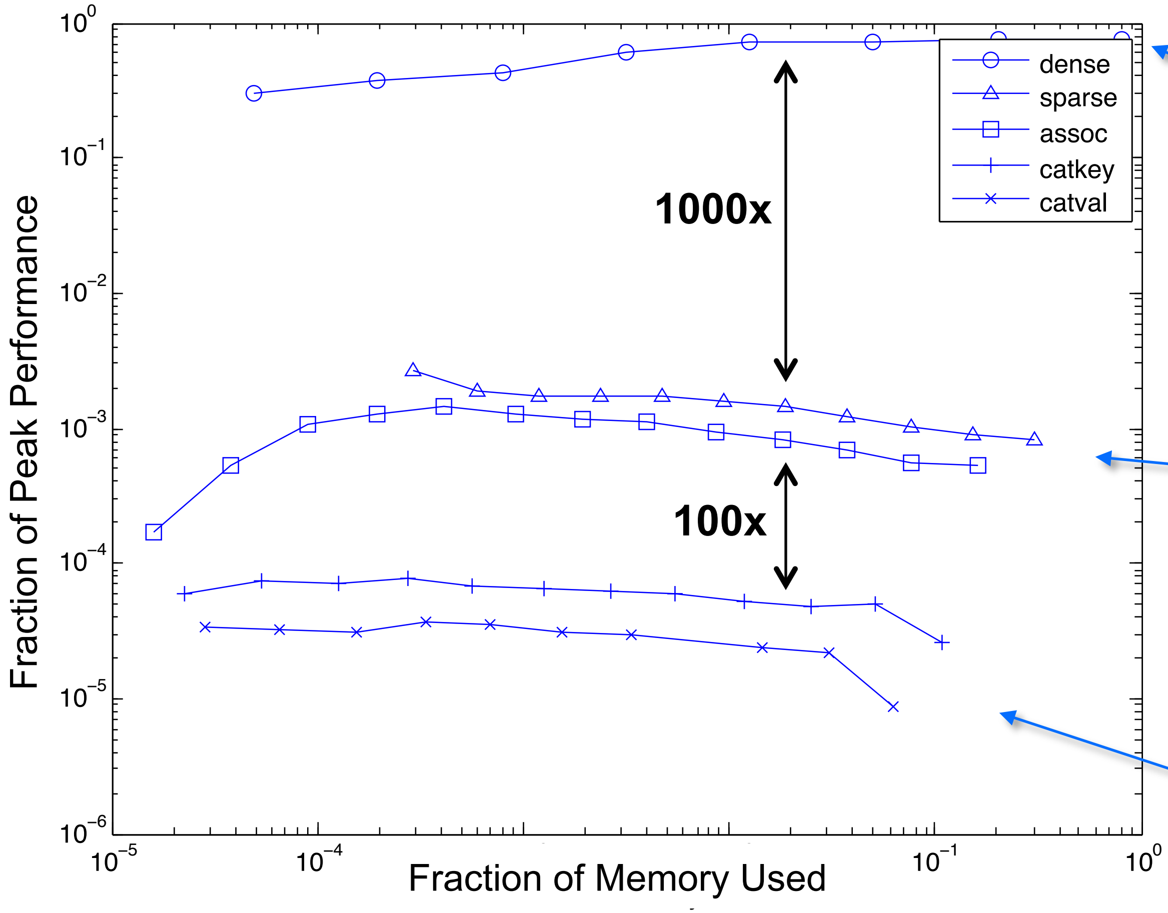
Key Kernels in Graph Analytics

Sparse matrix-dense vector multiplication (SpMV)	Sparse matrix-matrix multiplication (SpGEMM)	Breadth first search (BFS)
• Workhorse of sparse iterative methods (eigensolvers, CG, GMRES, etc.) • Signal processing for graphs	• Formation of correlation matrices • DNA sequence matching • Graph clustering	• Fundamental graph search algorithm • Graph 500 benchmark • Simple algorithm that stresses traditional architectures

Computational challenges

- **Sparsity of data**
- **Irregular data**
- **Lack of data locality (spatial and temporal)**

Performance Challenges in Graph Computations



Dense Linear Algebra
~100% Efficient.
What COTS is designed to do.

Sparse Linear Algebra
~0.1% Efficient.
What network analysis requires.

Sparse String Correlation
~0.001% Efficient.
What semantic analysis requires.

Performance for sparse linear algebra/graph operations significantly worse than dense linear algebra operations on COTS processors

Source: Jeremy Kepner, MITLL

Computational Architecture Choices

	ASIC	FPGA	Multicore 1	Multicore 2
Specific Architecture	65 nm CMOS IBM 10 LPe	Low Power Xilinx (Spartan 6, Samsung 45 nm)	GPGPU: NVIDIA Fermi	Intel Sandy Bridge
Method of Power characterization	Simulator	Simulator	• PAPI/NVML • LLMORE	• PAPI/RAPL • LLMORE

- Four specific architectures chosen
- Methodologies for power performance characterization of four architectures developed

Computational Architectures Comparison

	Programmability of kernels	Cost of repurposing	Expected power consumption	Parallelism
ASIC	🟦 Complex design; long fab time	🟧 Time consuming and expensive to refab	🟦 O(1 mW)	🟧 Can be designed to be highly parallel
FPGA	🟩 Requires RTL programming	🟩 Write new RTL code	🟩 O(100 mW)	🟩 Limited by number of gates
Nvidia Fermi	🟨 Requires CUDA programming	🟩 Write new CUDA code	🟧 ~200 W	🟨 Highly parallel due to 100s of CUDA cores
Intel Sandy Bridge	🟧 Many programming languages supported	🟦 Write new code	🟨 ~135 W	🟦 Limited by number of cores

🟦 Low 🟩 Medium 🟨 High 🟧 Very High

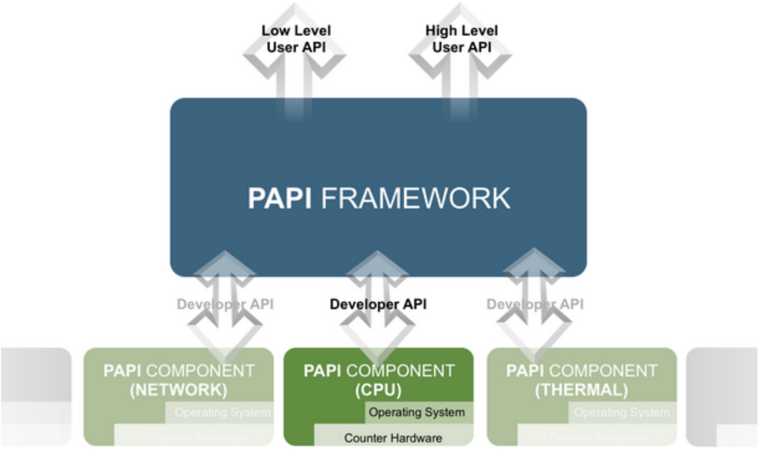
Each architecture has different advantages and disadvantages

Outline

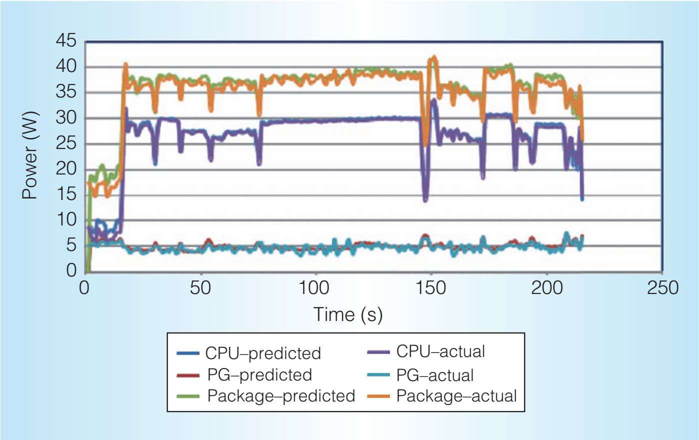
- Introduction
- Key Computational Kernels and Computational Architectures
- ➡ Results
- Exploration of Possible using LLMORE Simulator
- Other Key Benchmark Suites
- Summary and Future Work

Characterizing CPU Power/Performance with PAPI

- Performance Application Programming Interface (PAPI) provides access to hardware counters to monitor performance
  - Timing data
  - Cache hits/misses
  - Energy counters
    - Running Average Power Limit (RAPL) for SandyBridge CPU
    - NVIDIA Management Library (NVML) for NVIDIA GPGPU
- PAPI works across platforms
- Accurate power estimates from energy components



RAPL Estimates v Measured Power Usage



Rotem et. al., IEEE Micro, 2012

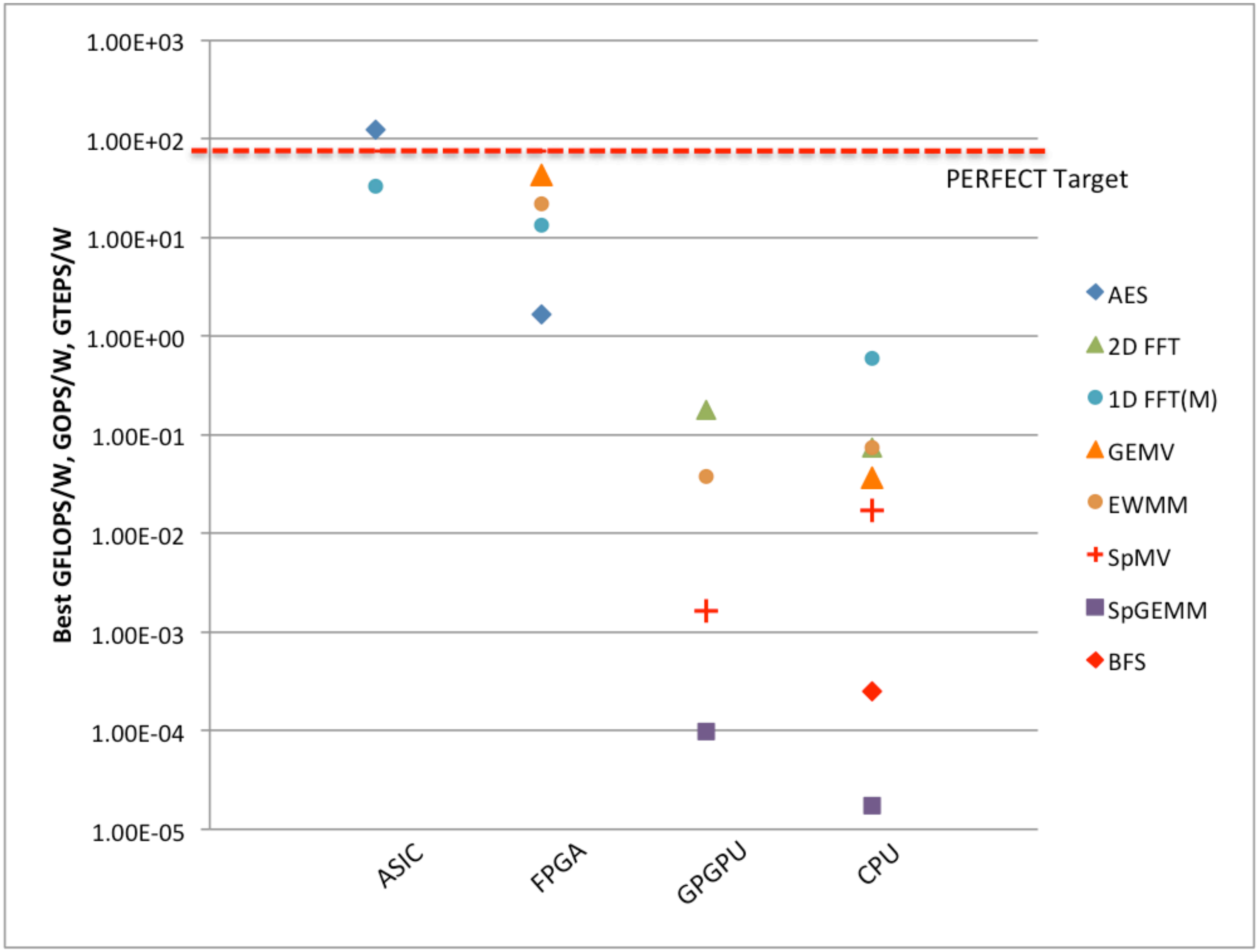
PAPI Power Measurements

- CPU Plan
  - Used PAPI to access RAPL
  - Hardware counters provide access to:
    - Package energy
    - DRAM energy
    - Energy of “Power Plane 0” (includes cores and caches)
  - Measurements:
    - In nanoJoules (nJ)
    - Sampled every microsecond
  - Averaged numerous trials to obtain accurate power estimates

- GPGPU Plan
  - Use NVML
  - Hardware Counters provide access to
    - Power (GPU, memory)
    - Temperature
  - Measurements:
    - Power in milliWatts (mW)
    - Temperature in Celsius (C)
  - Power Accuracy (Fermi)
    - Within +/- 5% current draw*

*NVML API Reference Manual, v 4.304.55, NVIDI, Oct. 2012

Preliminary Power Characterization Results



- 75 GFLOPS/W can be achieved with ASIC for certain kernels
- FPGAs are close to goal
- Sparse kernels perform orders of magnitude lower than dense kernels

- 75 GFLOPS/W is very challenging target for software programmable architectures

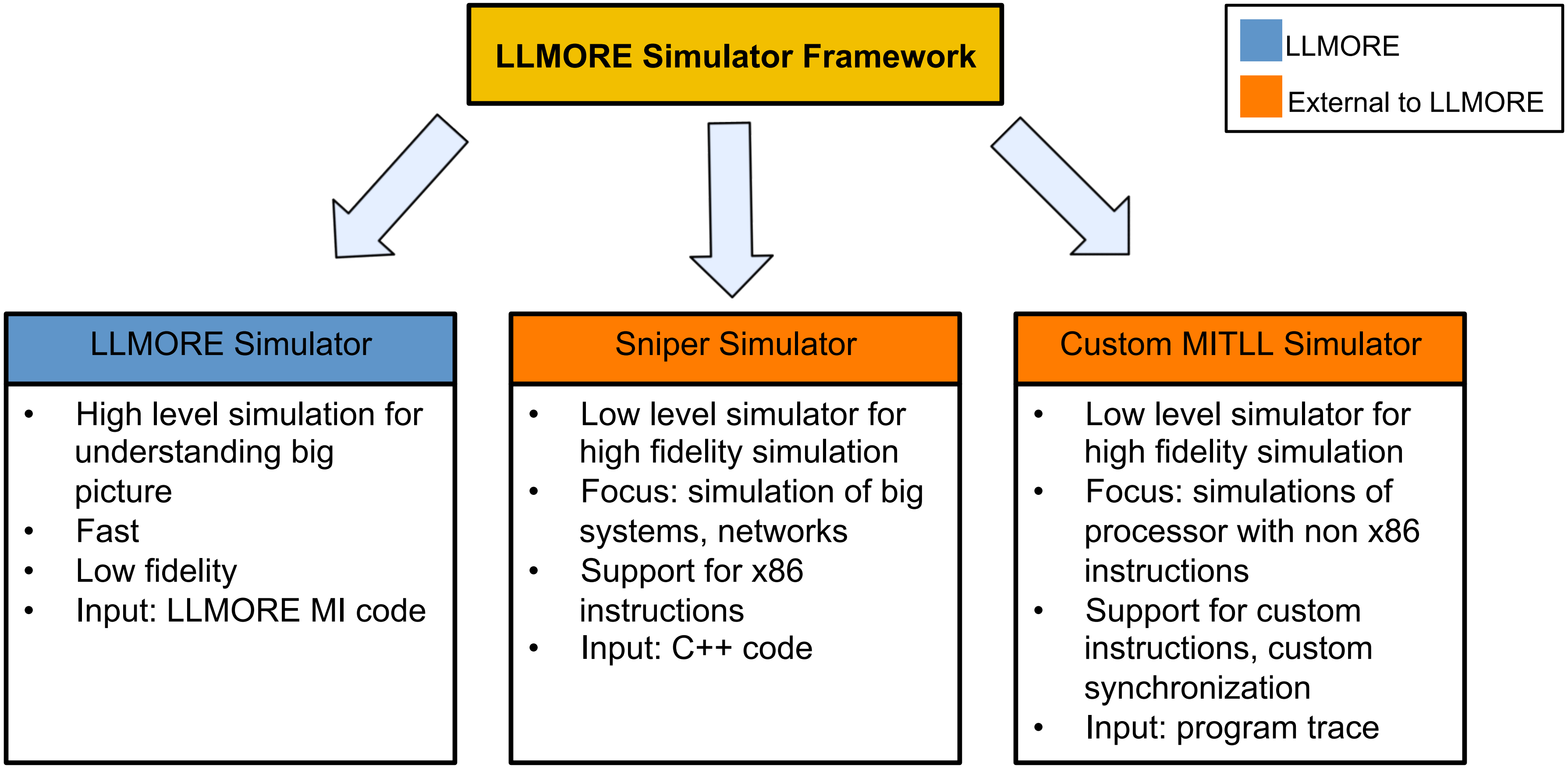
Exploration of the Possible

CPU and GPGPU Simulations using LLMORE

- MIT Lincoln Laboratory's Mapping and Optimization Runtime Environment (LLMORE) used for power and performance simulations of the possible
- LLMORE: parallel framework/environment for
  - Optimizing data to processor mapping for parallel applications
  - Simulating and optimizing new (and existing) architectures
  - Generating performance data (runtime, power, etc.)
  - Code generation and execution for target architectures
- LLMORE Simulations and PAKCK
  - Yield power and performance data for key computational kernels
  - Support for CPU and GPU architectures
  - Easy to add support for new architectures
    - Gives performance characterization of experimental architectures
    - Hybrid systems

LLMORE provides simulation support for key kernels on existing and future systems

LLMORE Simulator Framework



LLMORE Simulator Framework

🟦 LLMORE

🟧 External to LLMORE

LLMORE Simulator

- High level simulation for understanding big picture
- Fast
- Low fidelity
- Input: LLMORE MI code

Sniper Simulator

- Low level simulator for high fidelity simulation
- Focus: simulation of big systems, networks
- Support for x86 instructions
- Input: C++ code

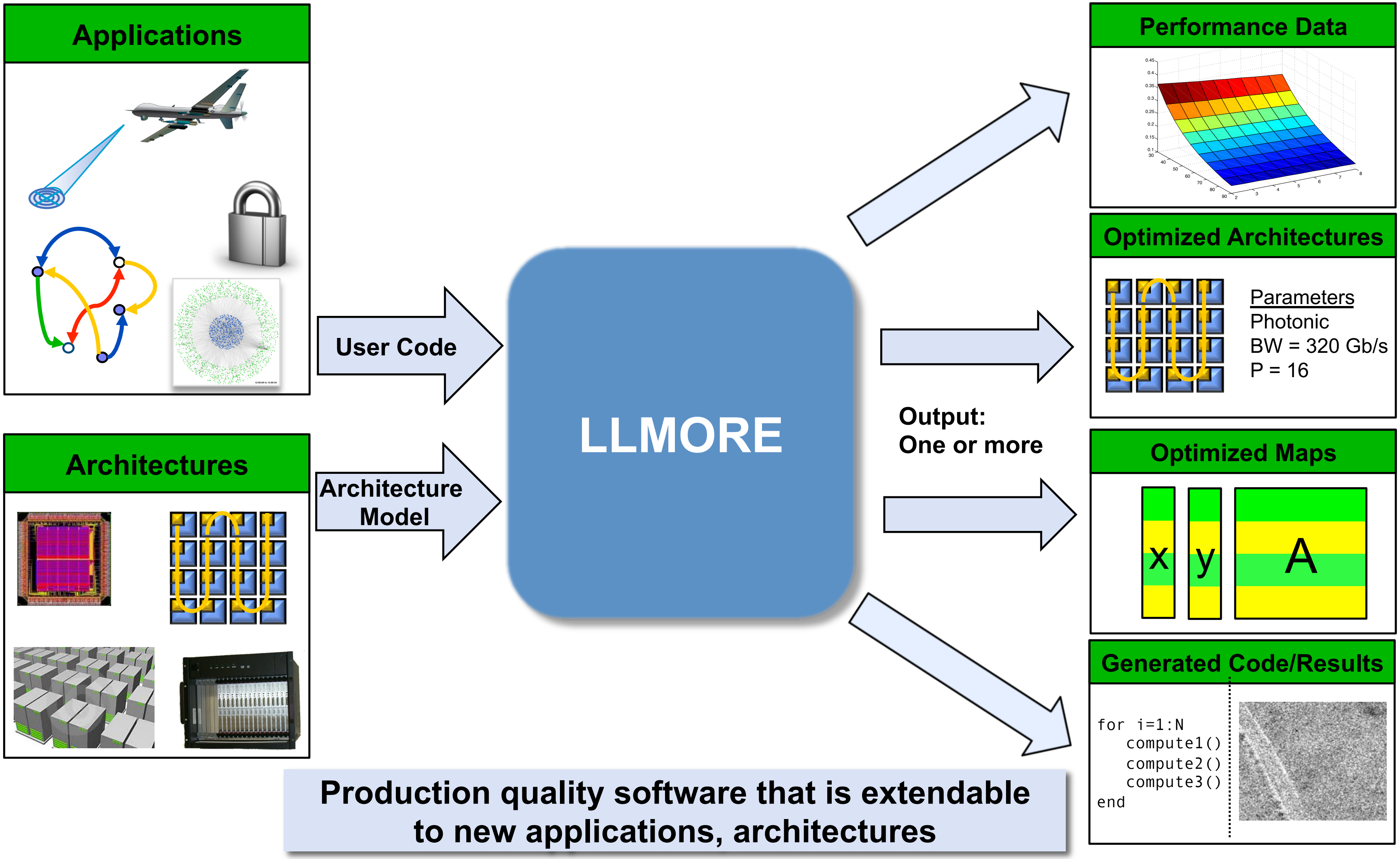
Custom MITLL Simulator

- Low level simulator for high fidelity simulation
- Focus: simulations of processor with non x86 instructions
- Support for custom instructions, custom synchronization
- Input: program trace

MI=machine independent

LLMORE interfaces to multiple simulators to support the analysis needs of different architectures.

LLMORE Overview



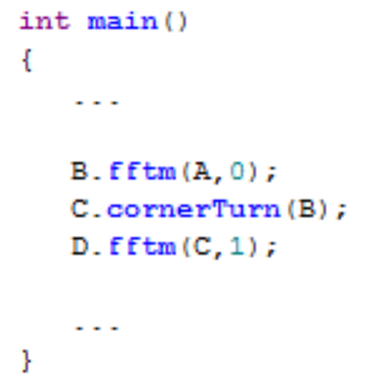
Parameters
Photonic
BW = 320 Gb/s
P = 16

```
for i=1:N
   compute1()
   compute2()
   compute3()
end
```

Production quality software that is extendable to new applications, architectures

Sample LLMORE Simulation – 2D FFT

Application

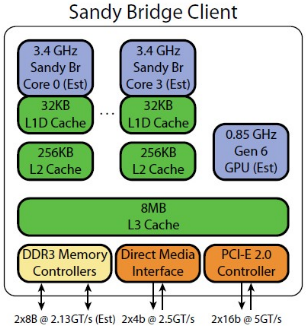


```
int main()
{
    ...

   B.fftm(A,0);
   C.cornerTurn(B);
   D.fftm(C,1);

    ...
}
```

Architecture



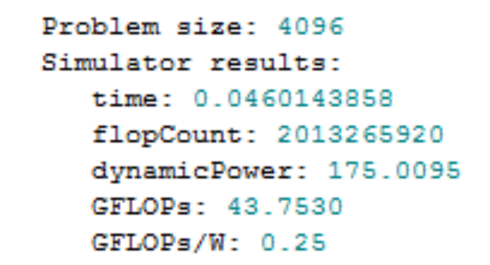
User Code

Architecture Model

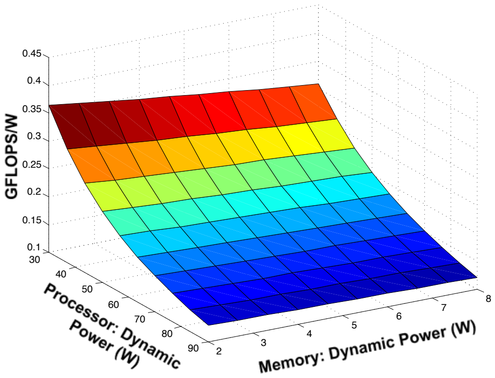
LLMORE

Output

Performance Data

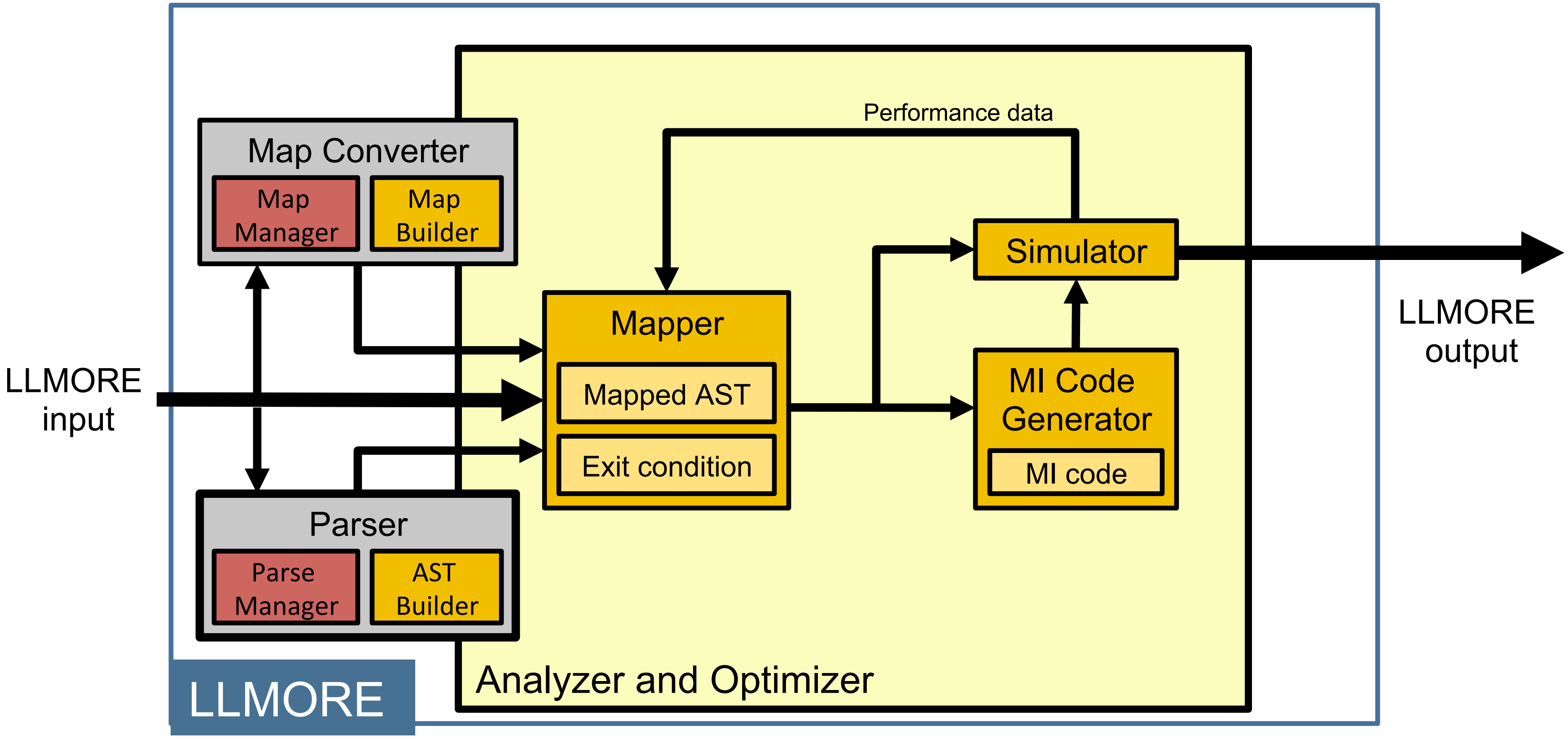


```
Problem size: 4096
Simulator results:
    time: 0.0460143858
    flopCount: 2013265920
    dynamicPower: 175.0095
    GFLOPs: 43.7530
    GFLOPs/W: 0.25
```



LLMORE simulates running 2D FFT on Sandy Bridge CPU and produces performance data

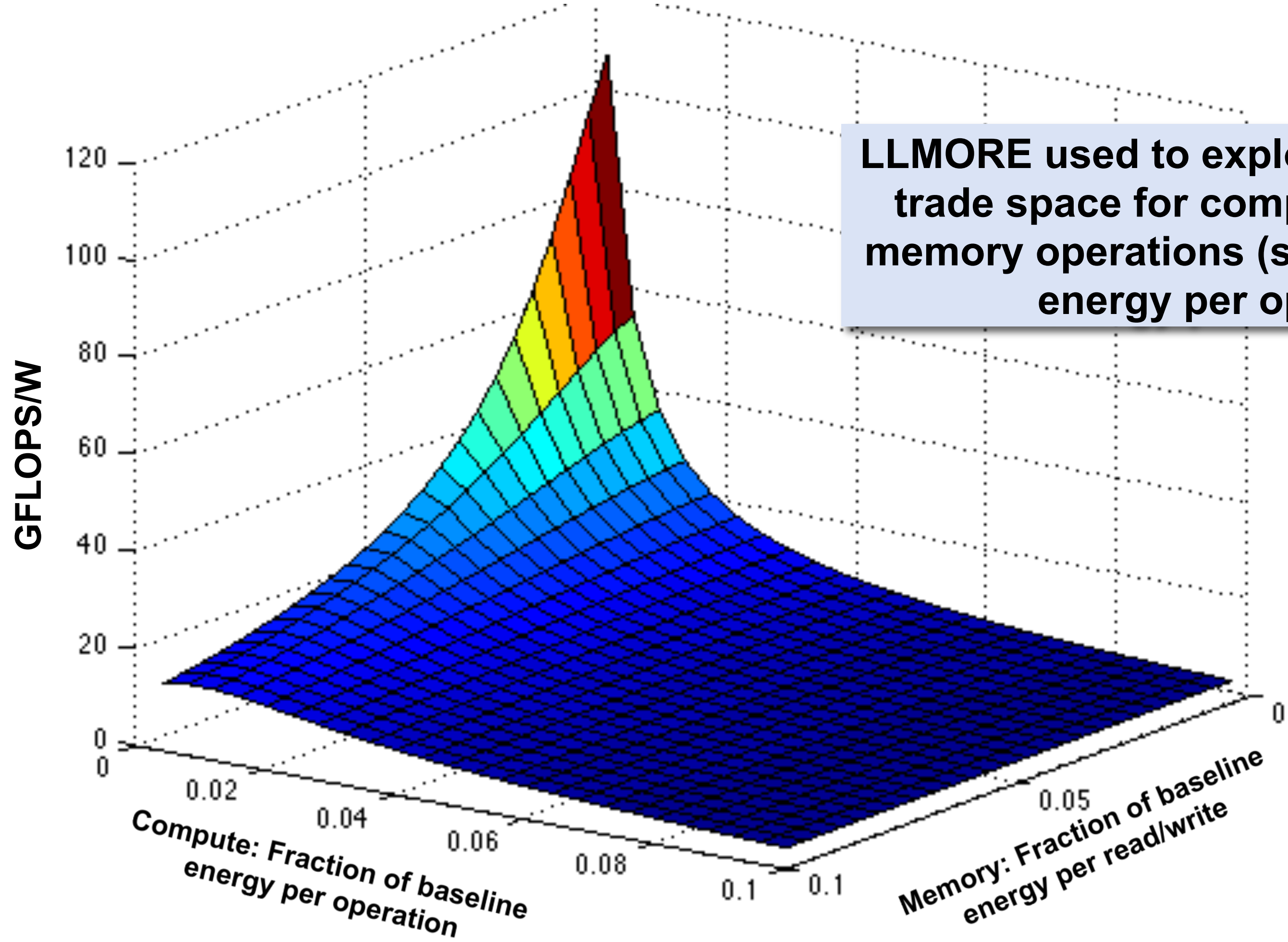
LLMORE Design: Detailed View



AST=abstract syntax tree, MI=machine independent

LLMORE requires coordinated interaction of multiple components

LLMORE: Exploration of the Possible



LLMORE used to explore energy trade space for compute and memory operations (scaling the energy per op)

Simulation indicates 50x-100x energy improvement needed in Intel Sandy Bridge to obtain 75 GFLOPS/W

Outline

- Introduction
- Key Computational Kernels and Computational Architectures
- Results
- Exploration of Possible using LLMORE Simulator
- Other Key Benchmark Suites
- Summary and Future Work

Other Key Benchmark Suites

HPCCHALLENGE HPCS

- High performance Linpack
- STREAM
- FFT
- RandomAccess
- Communication bandwidth and latency (b_eff)
- DGEMM & PTRANS

hpecchallenge

- Front-end stream processing kernels
- Back-end data analytics kernels
- Three scalable synthetic compact applications (SSCAs)
  - Pattern matching, graph analysis, synthetic aperture radar

GRAPH 500

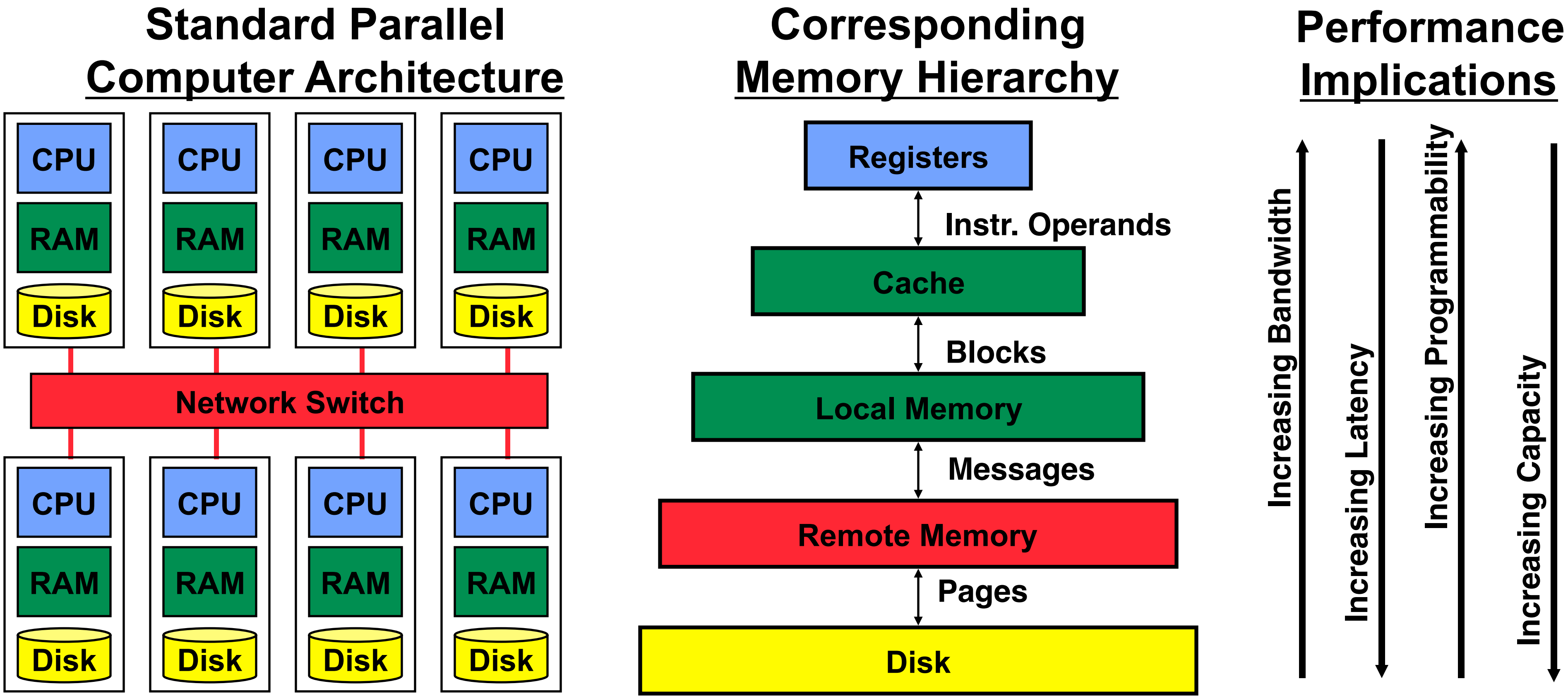
Graph 500 benchmark

- Data generator
- Breadth-first search

HPC Graph Analysis (Georgia Tech)

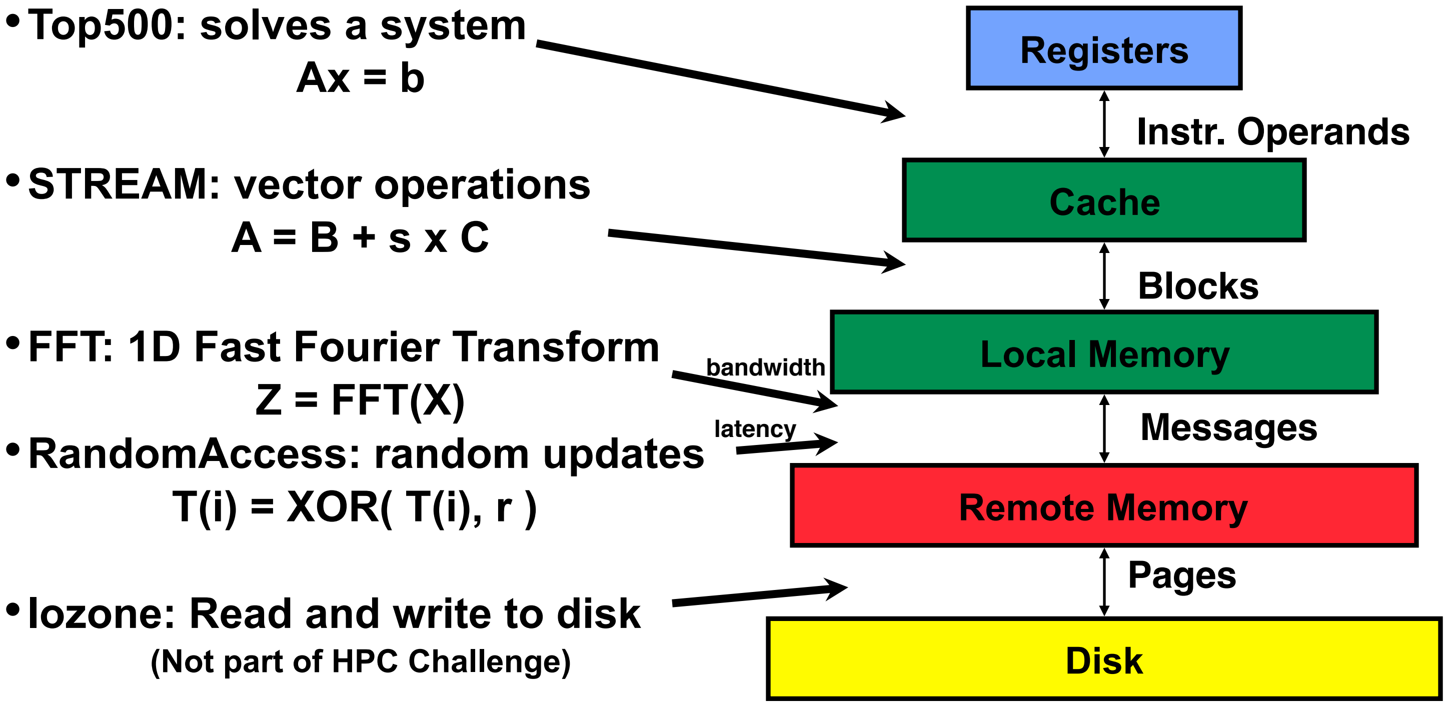
- Data generator
- Classify large sets
- Extract subgraphs
- Graph clustering
- graphanalysis.org

Parallel Computing Architecture Issues



- **Standard architecture produces a “steep” multi-layered memory hierarchy**
  - Programmer must manage this hierarchy to get good performance

HPC Challenge Benchmarks



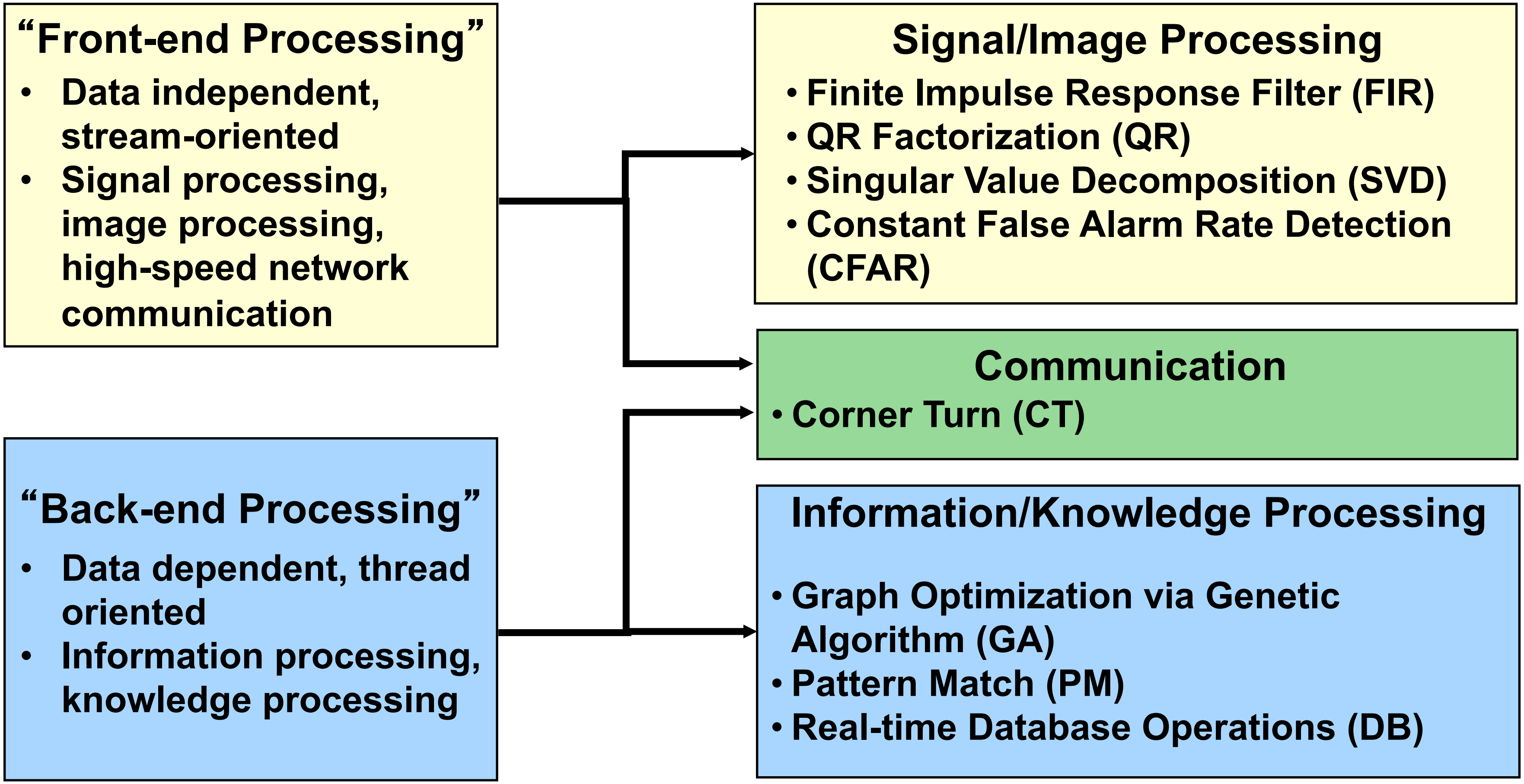
HPC Challenge
<u>Benchmark</u>

- **Top500: solves a system**
  Ax = b
- **STREAM: vector operations**
  A = B + s x C
- **FFT: 1D Fast Fourier Transform**
  Z = FFT(X)
- **RandomAccess: random updates**
  T(i) = XOR(T(i), r)
- **Iozone: Read and write to disk**
  (Not part of HPC Challenge)

Corresponding
<u>Memory Hierarchy</u>

- **HPC Challenge with Iozone measures this hierarchy**
- **Can determine whether each level of hierarchy is functioning properly**

HPEC Challenge: Kernel Benchmark Selection



Broad Processing Categories

"Front-end Processing"

- Data independent, stream-oriented
- Signal processing, image processing, high-speed network communication

"Back-end Processing"

- Data dependent, thread oriented
- Information processing, knowledge processing

Specific Kernels

Signal/Image Processing

- Finite Impulse Response Filter (FIR)
- QR Factorization (QR)
- Singular Value Decomposition (SVD)
- Constant False Alarm Rate Detection (CFAR)

Communication

- Corner Turn (CT)

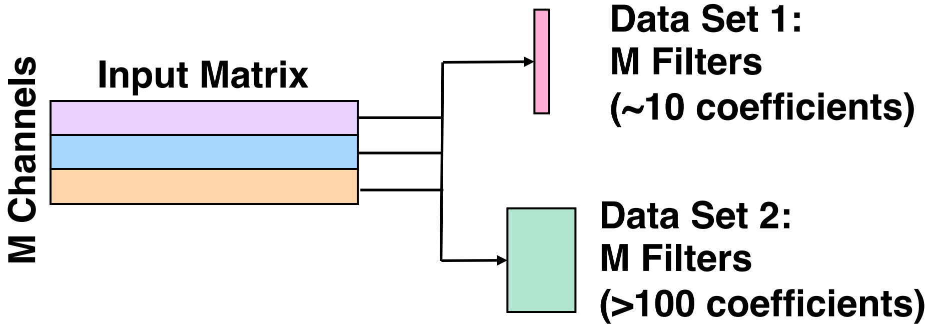
Information/Knowledge Processing

- Graph Optimization via Genetic Algorithm (GA)
- Pattern Match (PM)
- Real-time Database Operations (DB)

http://www.omgwiki.org/hpec/files/hpec-challenge/

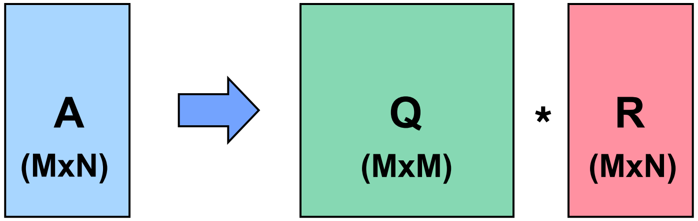
HPEC Challenge: Signal and Image Processing Kernels

FIR



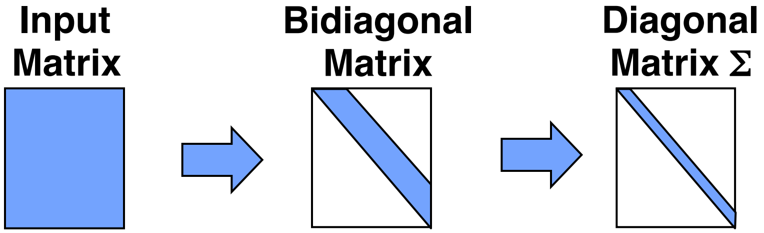
- Bank of filters applied to input data
- FIR filters implemented in time and frequency domain

QR



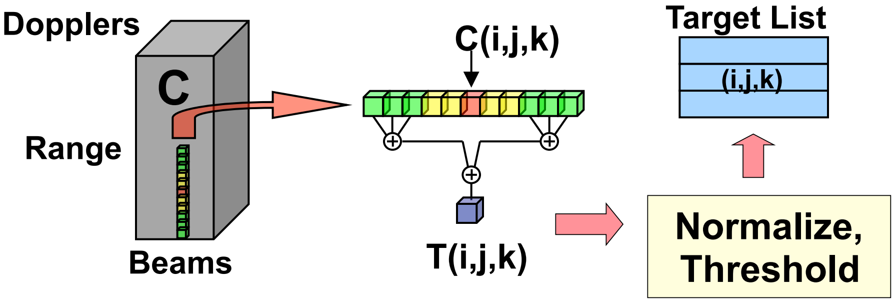
- Computes the factorization of an input matrix, A=QR
- Implementation uses Fast Givens algorithm

SVD



- Produces decomposition of an input matrix, $X=U\Sigma V^H$
- Classic Golub-Kahan SVD implementation

CFAR

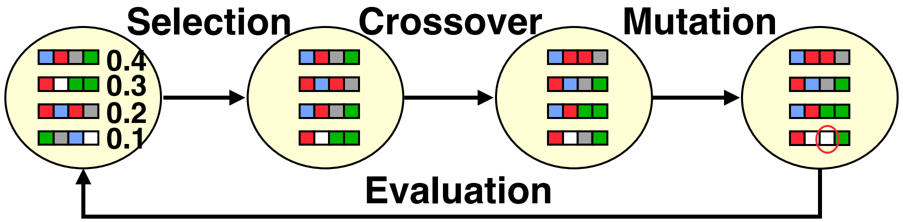


- Creates a target list given a data cube
- Calculates normalized power for each cell, thresholds for target detection

http://www.omgwiki.org/hpec/files/hpec-challenge/

HPEC Challenge: Information and Knowledge Processing Kernels

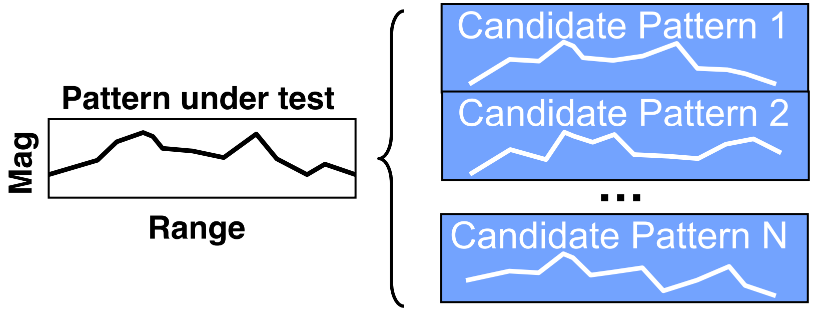
Genetic Algorithm



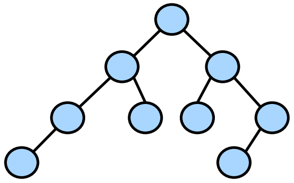
- Evaluate each chromosome
- Select chromosomes for next generation
- Crossover: randomly pair up chromosomes and exchange portions
- Mutation: randomly change each chromosome

Pattern Match

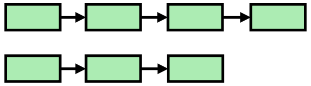
- Compute best match for a pattern out of set of candidate patterns
  - Uses weighted mean-square error



Database Operations



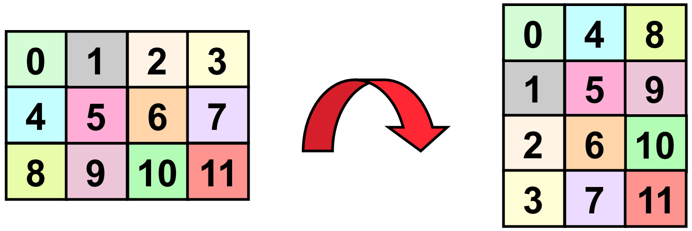
Red-Black Tree Data Structure



Linked List Data Structures

- Three generic database operations:
  - search: find all items in a given range
  - insert: add items to the database
  - delete: remove item from the database

Corner Turn



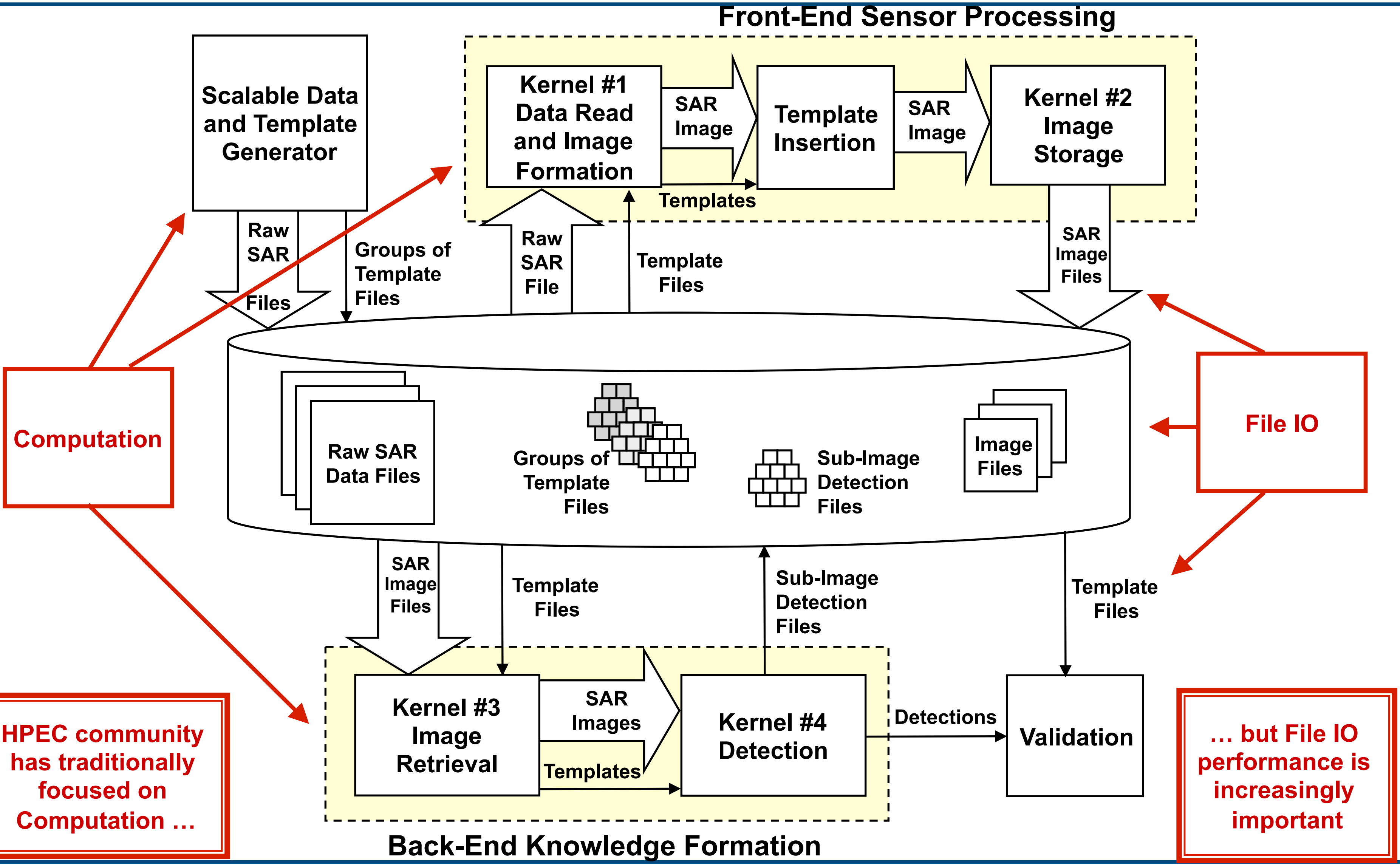
0	1	2	3
4	5	6	7
8	9	10	11

0	4	8
1	5	9
2	6	10
3	7	11

- Memory rearrangement of matrix contents
  - Switch from row to column major layout

http://www.omgwiki.org/hpec/files/hpec-challenge/

HPEC Challenge SSCA#3: SAR System Architecture



HPEC community has traditionally focused on Computation …

… but File IO performance is increasingly important

http://www.omgwiki.org/hpec/files/hpec-challenge/

Summary and Future Work

Summary

- Major finding: DARPA is targeting ASIC-levels of computational efficiency applied to programmable computational architectures
- PAKCK results show this is a challenging goal to achieve
- PAKCK has quantified the gap between current programmable computational architectures and DARPA goal for DoD-relevant application kernels

Future Work

- Characterize performance bottlenecks on Sandy Bridge for SpMV and SpGEMM
- Extend LLMORE to simulating other device technologies