Analysis and Mapping of Sparse Matrix Computations

Nadya Bliss & Sanjeev Mohindra
MIT Lincoln Laboratory

Varun Aggarwal & Una-May O'Reilly
MIT Computer Science and AI Laboratory

September 19th, 2007

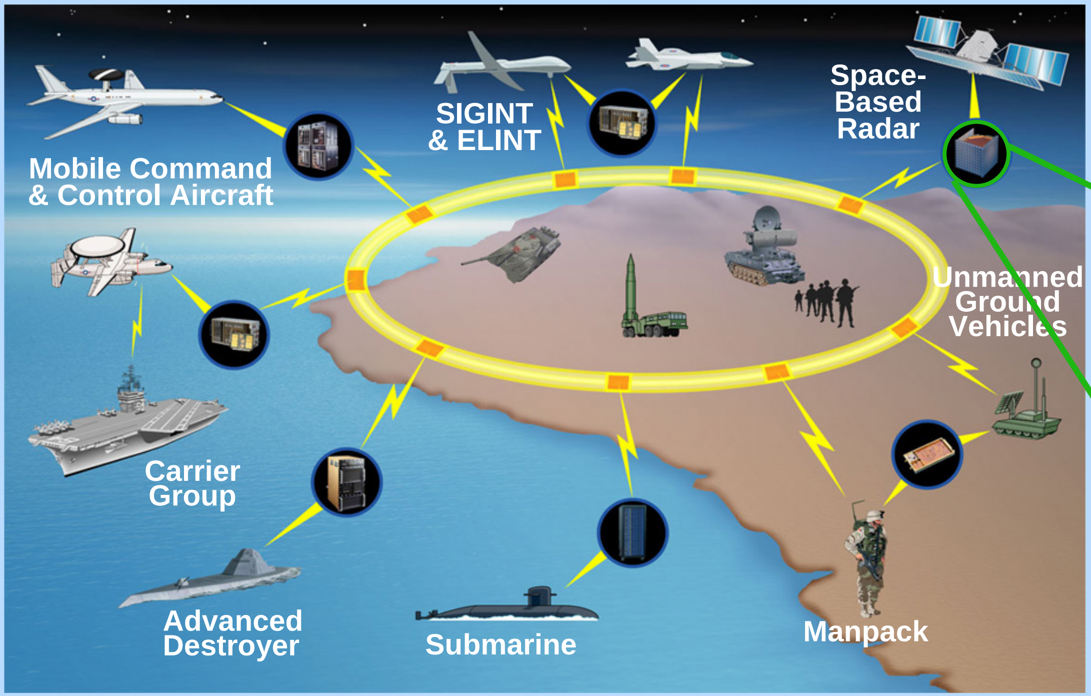
This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Outline

- **Introduction**
- Sparse Mapping Challenges
- Sparse Mapping Framework
- Results
- Summary

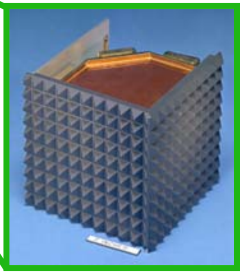
Emerging Sensor Processing Trends

Highly Networked System-of-Systems Sensors and Computing Nodes



Processing Challenges

- Small Platforms
- Smart Sensors
- Scalable Sensors Networks



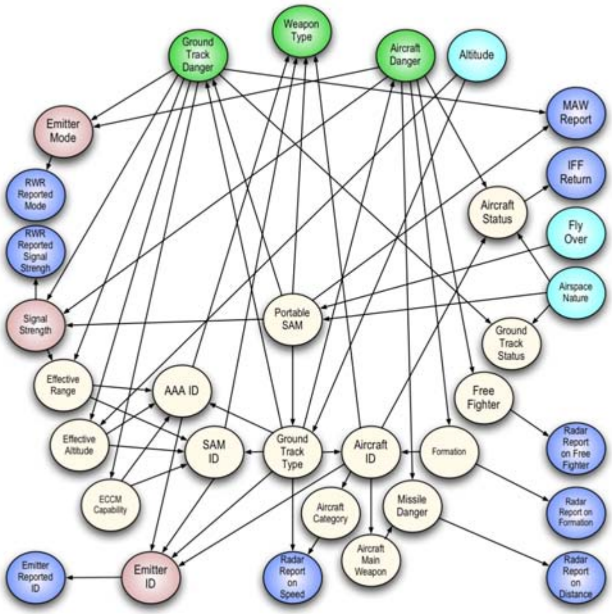
Enabling Technologies

- Extreme Form Factor Processors
- Knowledge-based Algorithms
- Efficient Algorithm-to-Architecture Implementations

Rapid growth in size of data and complexity of analysis are driving the need for real-time knowledge processing at sensor front end.

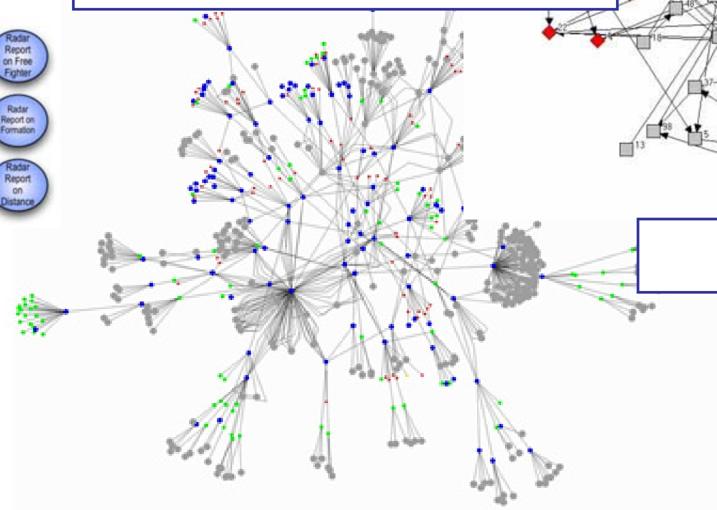
Knowledge Processing & Graph Algorithms

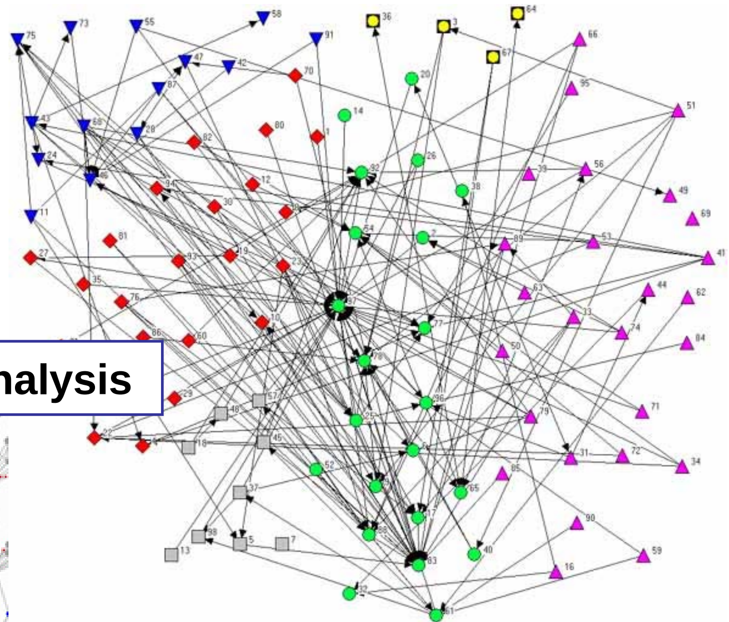
Many knowledge processing algorithms are based on graph algorithms



Target Identification

Social Network Analysis





Anomaly Detection

Algorithmic Techniques:
- **Bayesian Networks**
- **Neural Networks**
- **Decision Trees**

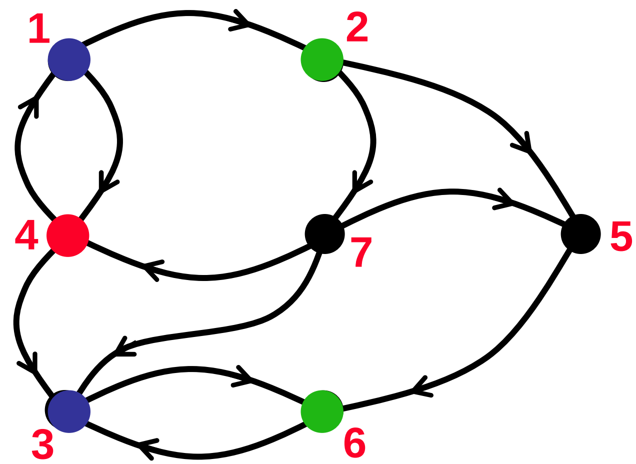
...

Graph-Sparse Duality

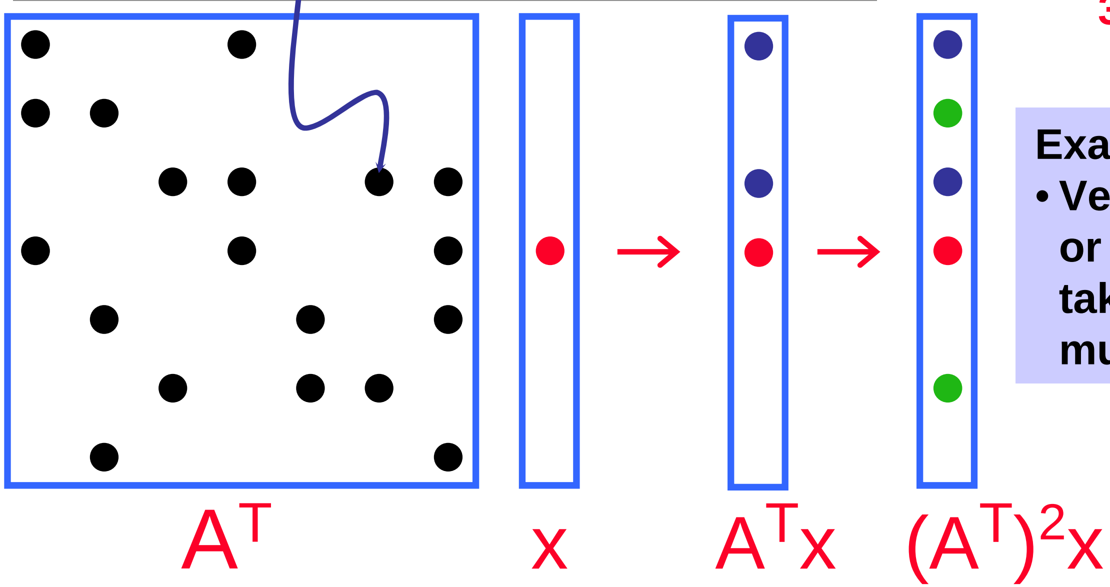
Many graph algorithms can be expressed as *sparse matrix* computations

Graph preliminaries
A graph G = (V,E) where
- **V = set of vertices**
- **E = set of edges**

Graph G:



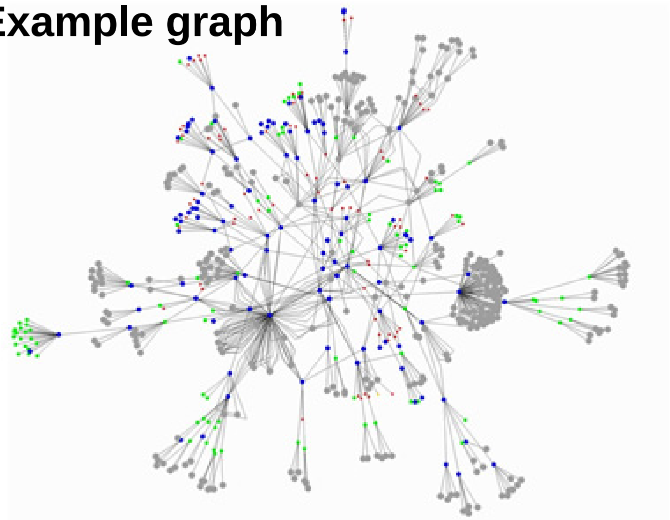
Adjacency matrix representation:
- **Non-zeros entry A(i,j) where there exists an edge between vertices i and j**



Example operation:
- **Vertices reachable from vertex v in N or less steps can be computed by taking A to the Nth power and multiplying by a vector representing v**

Motivating Example
-Computing Vertex Importance-

Example graph

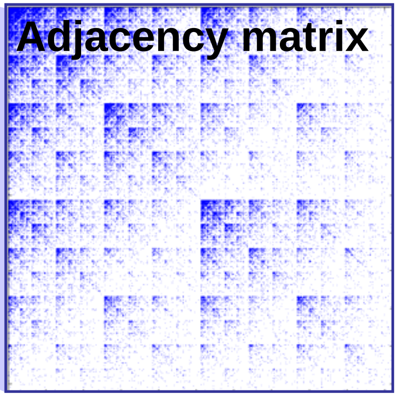


Common computation:
- **Vertex/edge importance**
- **Graph/sparse duality: matrix multiply**

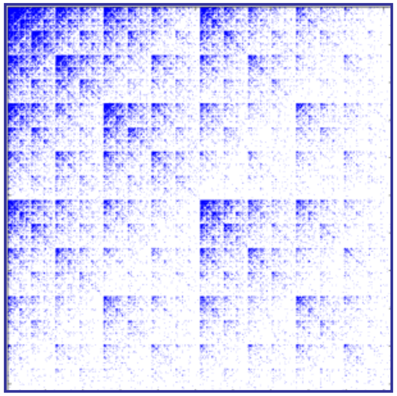
Applications in:
- **Social Networks**
- **Biological Networks**
- **Computer Networks and VLSI Layout**
- **Transportation Planning**
- **Financial and Economic Networks**

- **Matrix multiply is computed for each vertex**
- **Must be recomputed if graph is dynamic (changing connections between nodes)**
- **Current typical efficiency: 0.001 of peak performance**

Adjacency matrix



X



Sparse computations are <0.1% efficient.

Outline

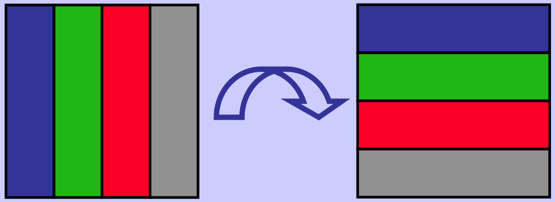
- Introduction
- **Sparse Mapping Challenges**
- Sparse Mapping Framework
- Results
- Summary

Mapping of Dense Computations

Common dense array distributions:

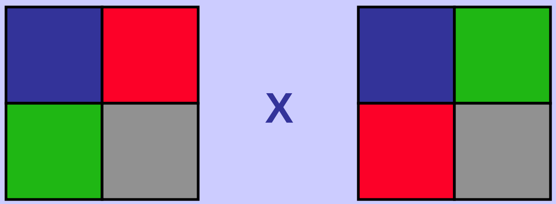
1D Block

eg. FFT



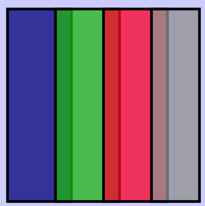
2D Block

eg. Matrix multiply



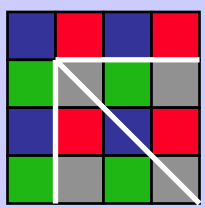
Block overlap

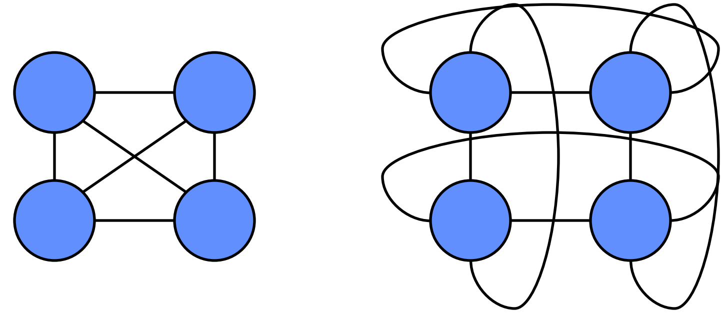
eg. Convolution



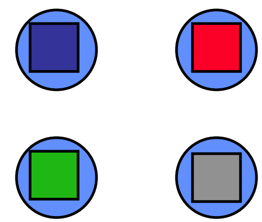
Block cyclic

eg. LU decomposition





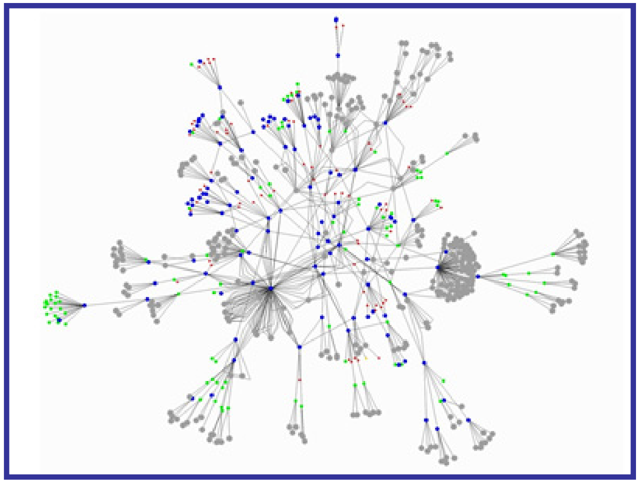
Well understood communication patterns

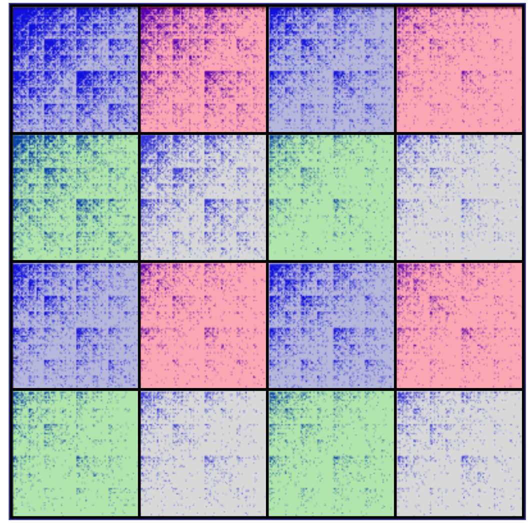


Good load balancing

Regular distributions allow for efficient mapping of dense computations

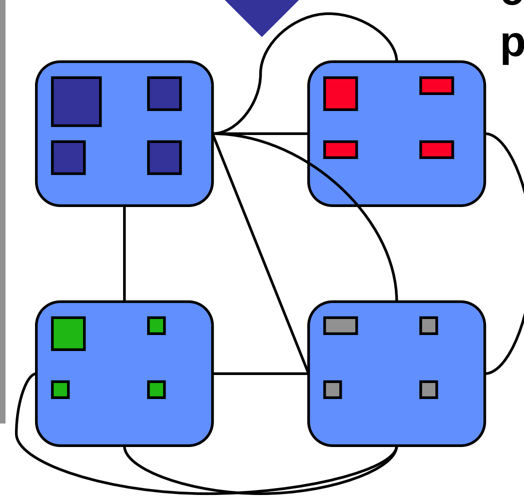
Mapping of Sparse Computations





Block cyclic mapping is commonly used for sparse matrices

Data and computation are poorly balanced



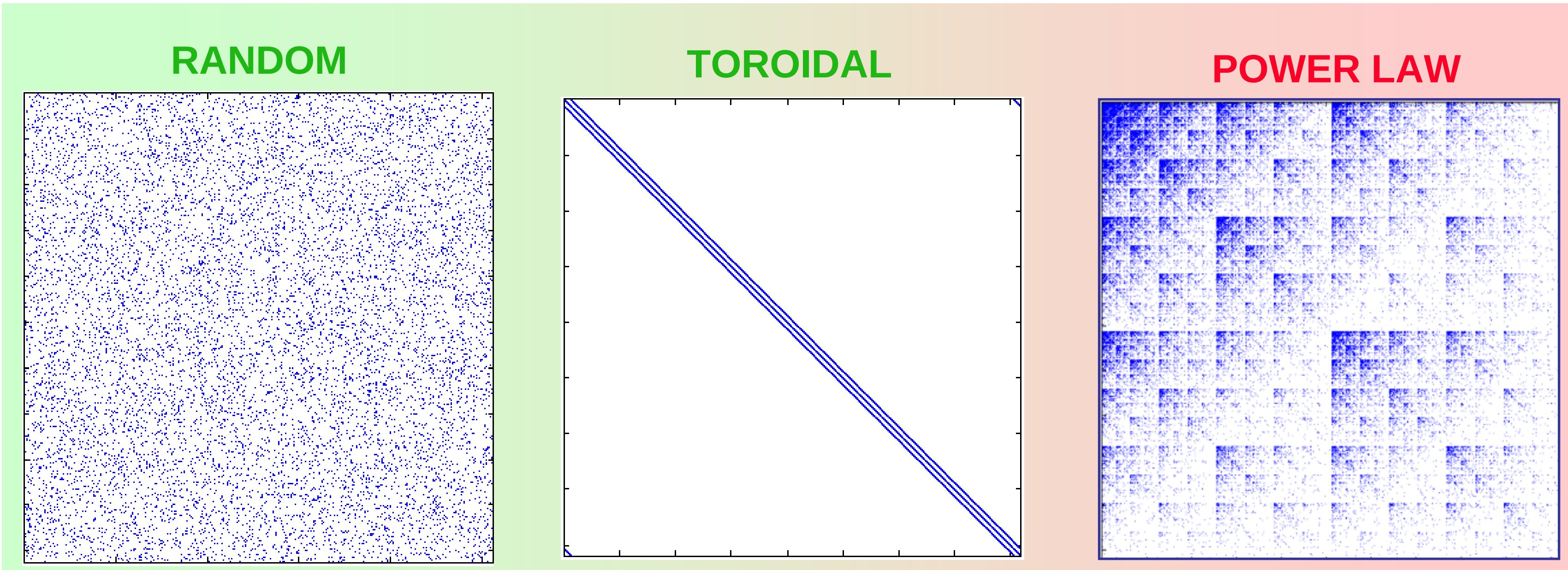
Communication pattern is irregular and fine-grained

Key Challenges

- *Fine-grained* computation
- *Fine-grained* communication
- *Co-optimization* of computation and communication at the fine-grain level

Common Types of Sparse Matrices

Sparsity structure of the matrix has significant impact on mapping

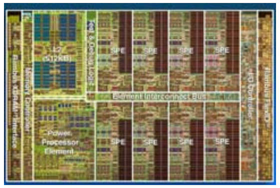


Increasing load balancing complexity

Outline

- Introduction
- Sparse Mapping Challenges
- **Sparse Mapping Framework**
- Results
- Summary

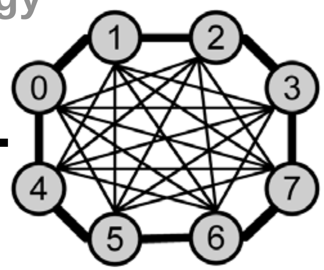
Dense Mapping Framework

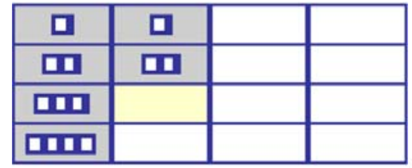


Coarse-grained machine model with all-to all topology

```
n_cpus
cpu_rate
mem_rate
net_rate
cpu_latency
```

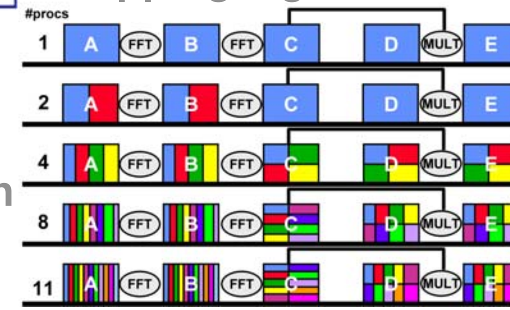
+

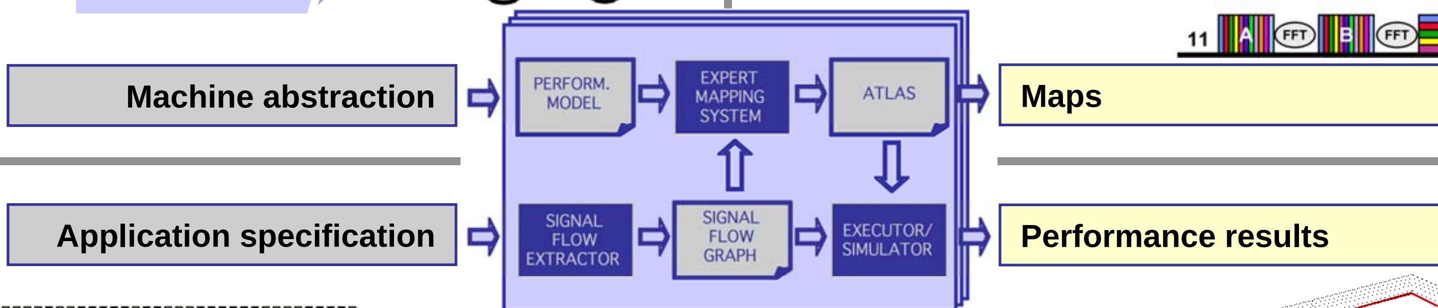


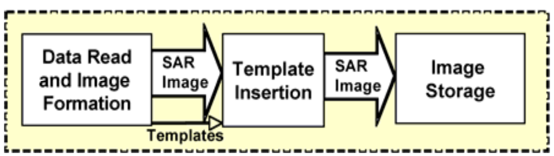


Regular distribution

Heuristic dynamic programming mapping algorithm

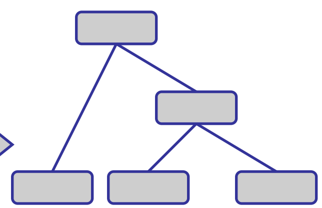






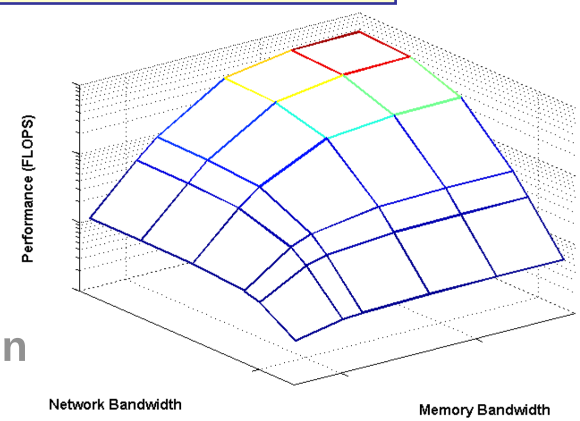
APPLICATION

Coarse-grained program analysis



SIGNAL FLOW GRAPH

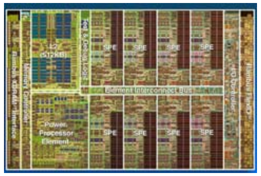
Performance prediction and processor characterization

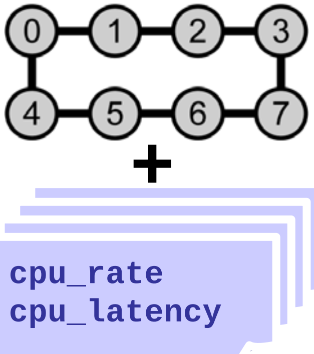


Bliss, et al. Automatic Mapping of HPEC Challenge Benchmarks, HPEC 2006.
Travinin, et al. pMapper: Automatic Mapping of Parallel MATLAB Program, HPEC 2005.

Sparse Mapping Framework

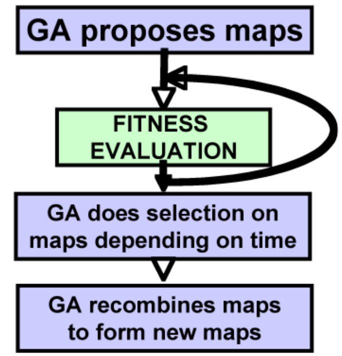
MACHINE ABSTRACTION

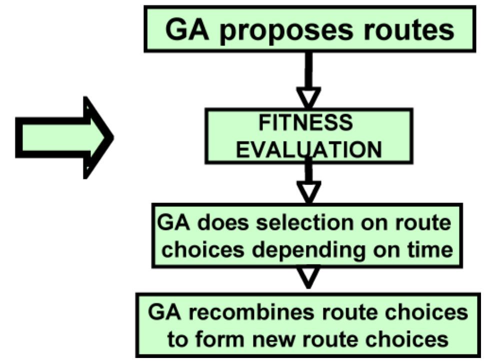




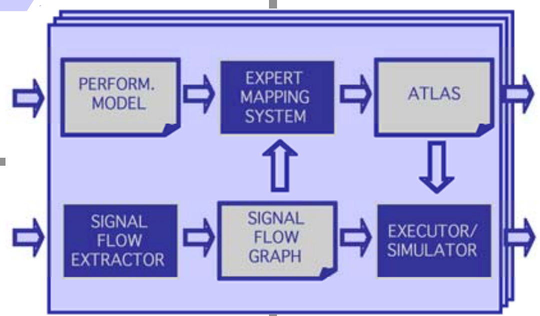
Detailed, topology-true machine model

MAPPING ALGORITHM



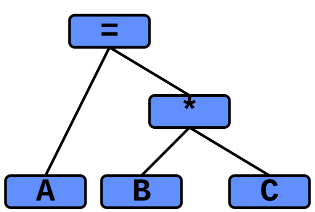


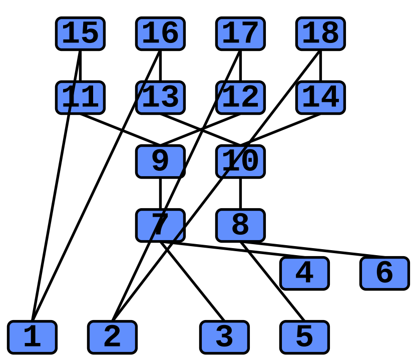
Nested GA for mapping and routing



Fine-grained program analysis

Support for irregular distributions





1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

PROGRAM ANALYSIS

OUTPUT MAPS

Sparse Mapping Framework

MACHINE ABSTRACTION

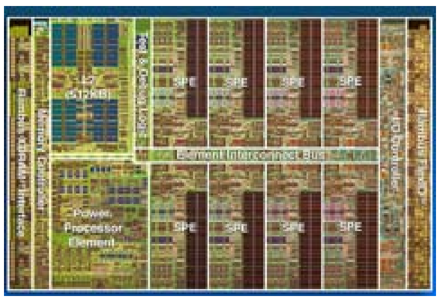
Latency and Bandwidth as a Graph

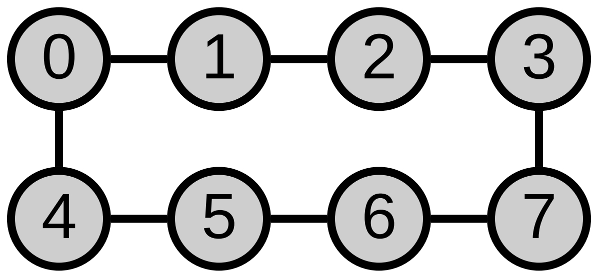
- $L_{ij,\, i \neq j}$ = latency between nodes *i* and *j*
- $B_{ij,\, i \neq j}$ = bandwidth between nodes *i* and *j*
- L_{ii} = memory latency
- B_{ii} = memory bandwidth
- Model preserves topology information

CPU, etc as an Array

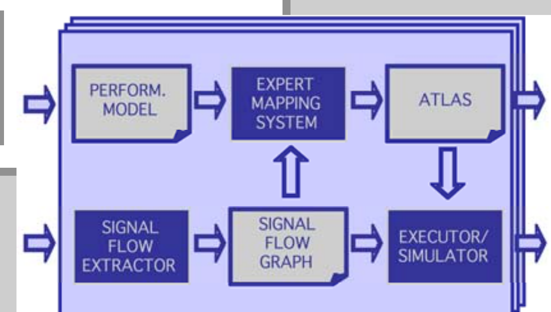
```
cpu_rate
cpu_latency
```

Parameters stored per processor - heterogeneity



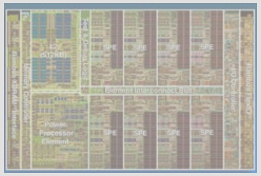


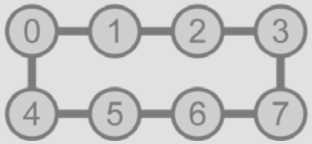
Detailed, topology-true machine abstraction allows for accurate *modeling* of fine-grain operations



Sparse Mapping Framework

MACHINE ABSTRACTION

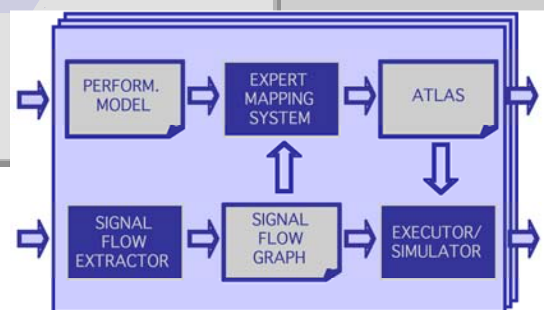




+

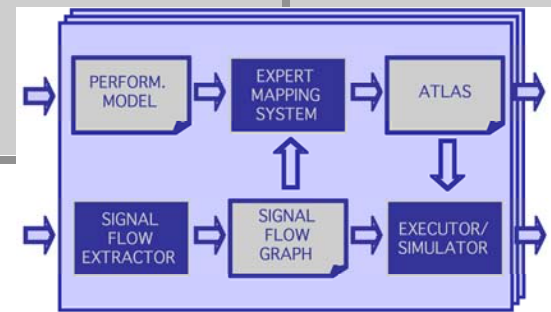
```
cpu_rate
cpu_latency
```

Detailed, topology-true machine model

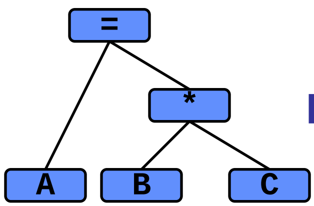


PROGRAM ANALYSIS

Sparse Mapping Framework



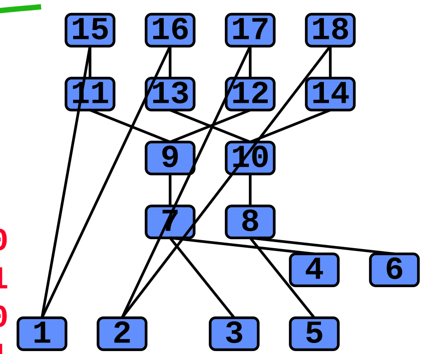
FINE-GRAINED PROGRAM ANALYSIS



memory

```
...
Proc 3: mem_op(24) - B
Proc 3: mem_op(24) - C
Proc 2: cpu_op(24) - B+C
Proc 3: cpu_op(24) - B+C
Proc 2: mem_op(24) - temp
Proc 3: mem_op(24) - temp
Proc 2: net_op(12) - to Proc 0
Proc 2: net_op(12) - to Proc 1
Proc 3: net_op(12) - to Proc 0
Proc 3: net_op(12) - to Proc 1
...
```

computation

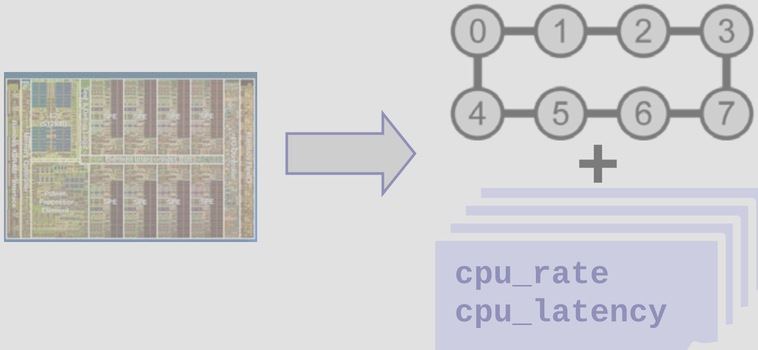


communication

FGSFG allows for accurate *representation* of fine grain computations on a detailed machine topology.

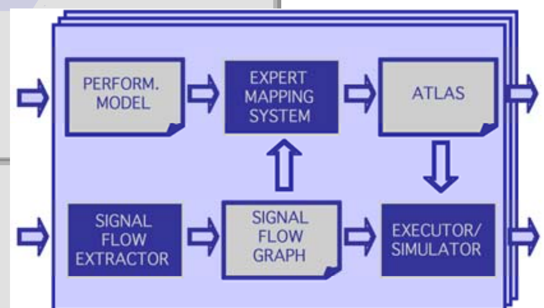
Sparse Mapping Framework

MACHINE ABSTRACTION

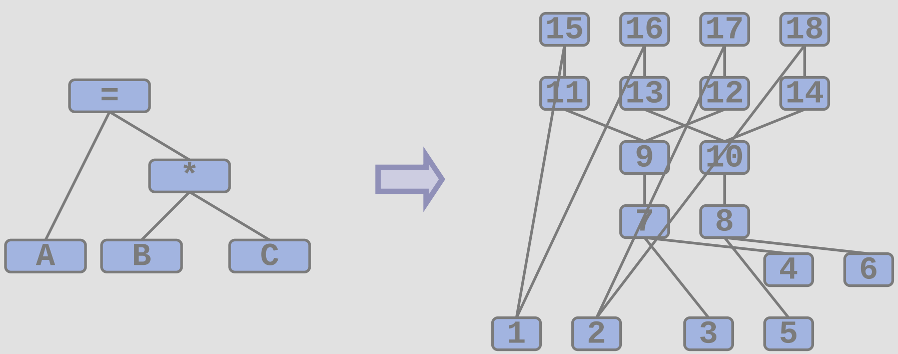


Detailed, topology-true machine model

MAPPING ALGORITHM



Fine-grained program analysis

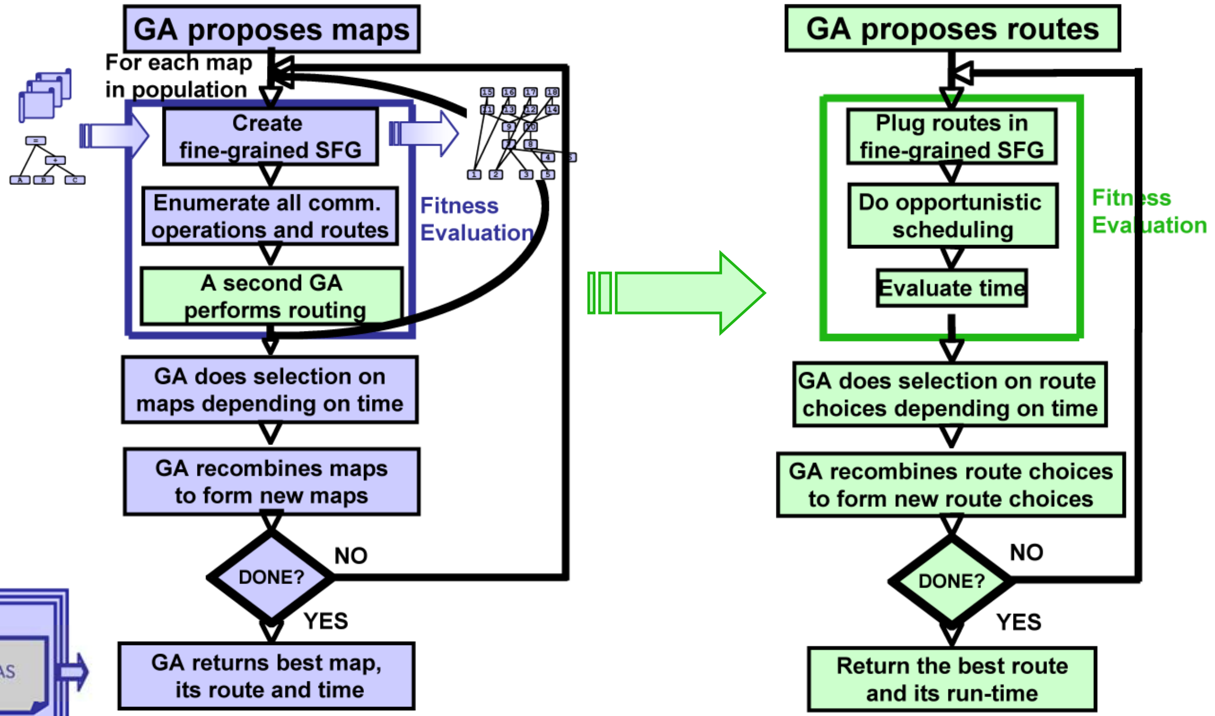


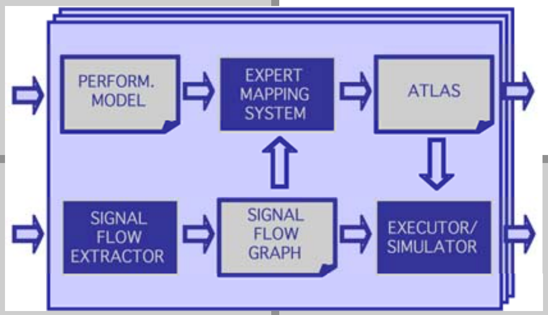
PROGRAM ANALYSIS

Sparse Mapping Framework

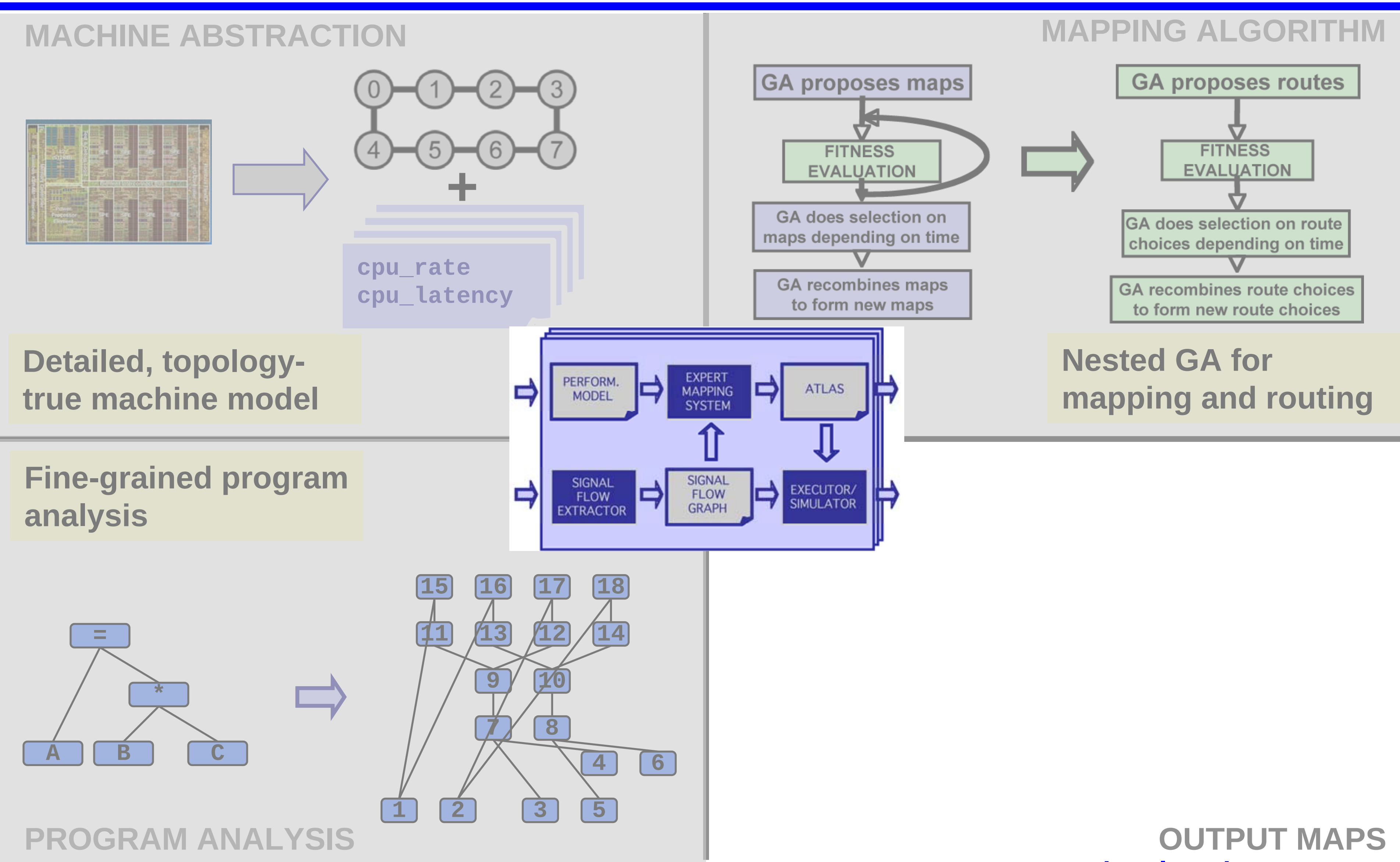
MAPPING AND ROUTING ALGORITHM

Combinatorial nature of the problem makes it well suited for an approximation approach: nested genetic algorithm (GA)

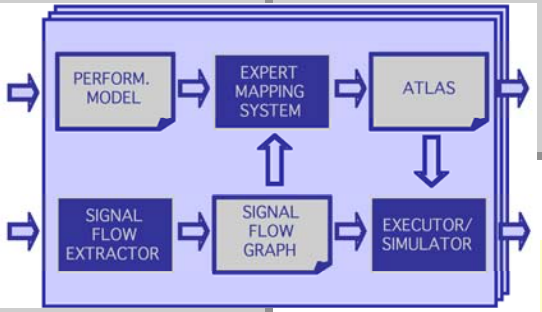




Sparse Mapping Framework



Sparse Mapping Framework



DATA DISTRIBUTIONS

Irregular data distributions allow exploration of fine-grained mapping search space of sparse computations

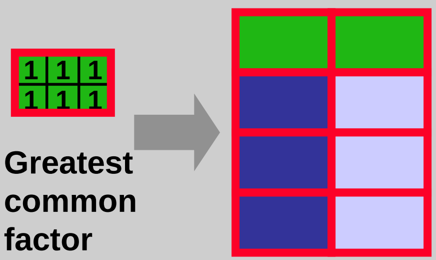
1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0

Block Size **Processor Grid**



Greatest common factor

```
map definition:
grid: 4x2
dist: block
proc list:
[0 1 1 1 0 2 2 2]
```

Allow processor rank repetition in the map processor list

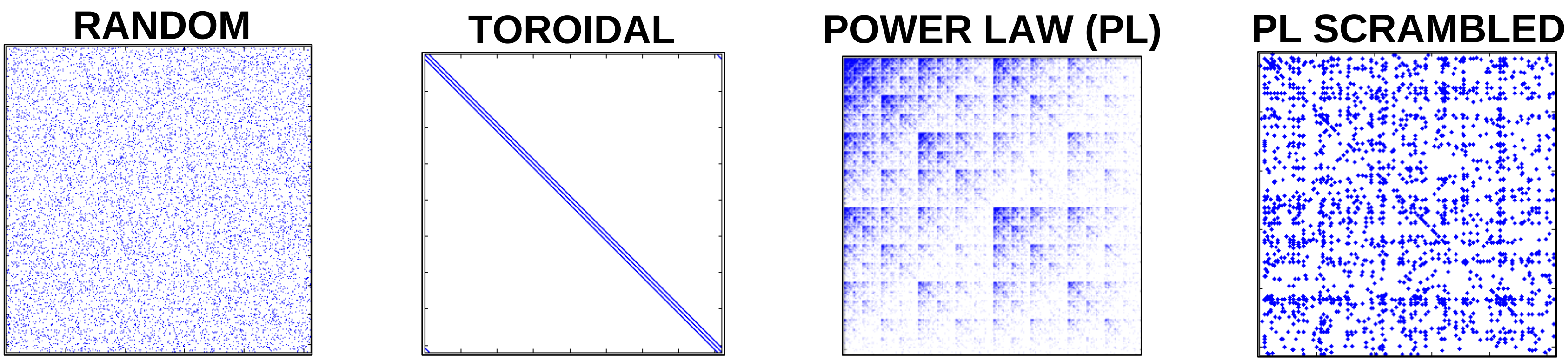
Standard redistribution and indexing techniques apply

Outline

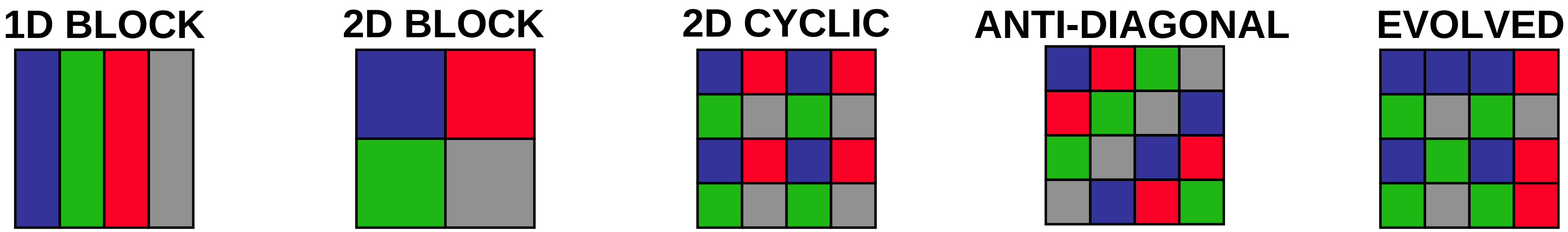
- Introduction
- Sparse Mapping Challenges
- Sparse Mapping Framework
- **Results**
- Summary

Experiments

MATRIX TYPES:



DISTRIBUTIONS:



Evaluate the sparse mapping framework on various matrix types and compare with performance of regular distributions

Results: Performance

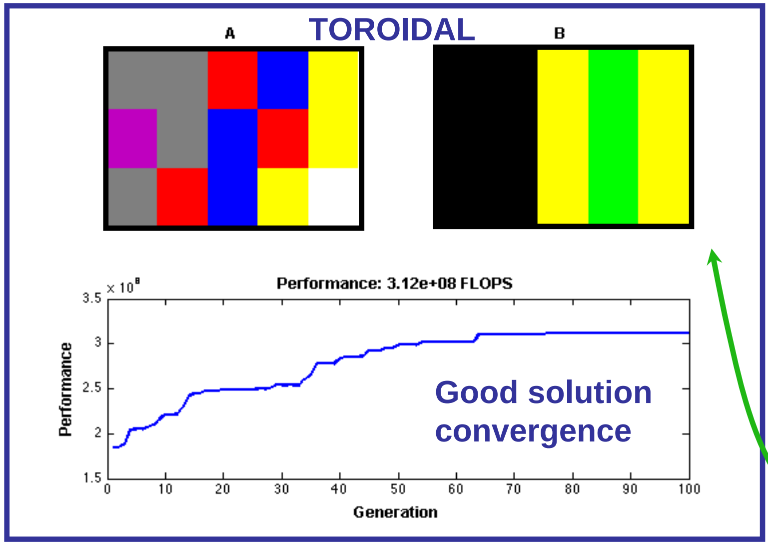
Matrix Type	1D Block	2D Block	2D Block Cyclic (FLOPS)	Anti-Diag Cyclic	Evolved
Random Sparse	0.8	0.8	1 ($3.3*10^7$)	2.6	11
Power Law	1.3	.6	1 ($1.0*10^8$)	2.7	18
Power Law Scrambled	1.4	1.4	1 ($2.0*10^7$)	3	17
Toroidal	2.5	1.4	1 ($3.3*10^6$)	8.5	94

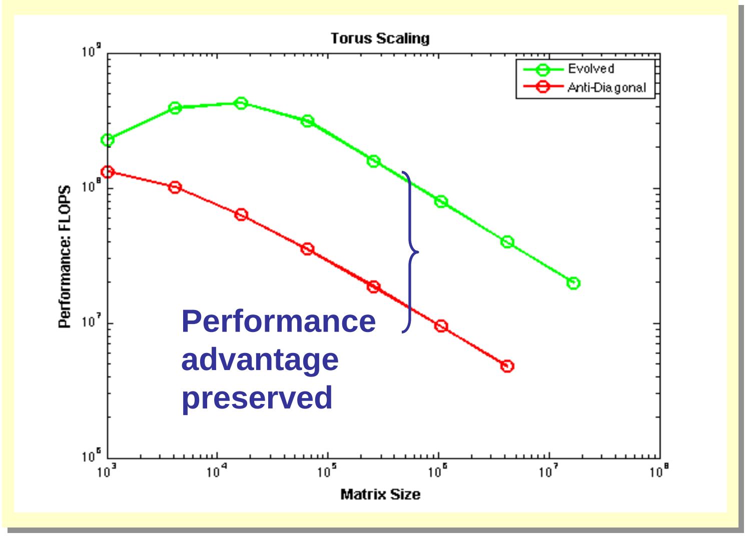
Experiment details:
- Results relative to 2D block cyclic distribution
- Machine model: 8 processor ring with 256 GB/sec bandwidth
- Matrix size: 256x256
- Number of non-zeros: 8*256

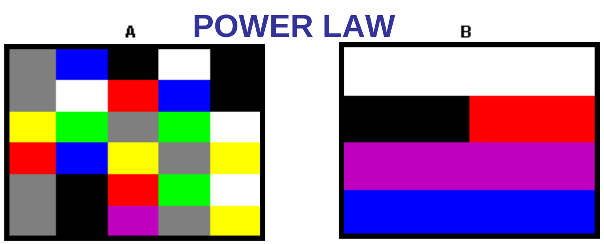
Sparse mapping framework outperforms all other distributions on all matrix types

Results: Maps and Scaling

Map evolved for a 256x256 matrix applied to 32x32 to 4096x4096







Simpler mapping for matrix B characteristic of parallel matrix multiply algorithm

Sparse mapping framework exploits both matrix structure and algorithm properties

Summary

- **Digital array sensors are driving the need for knowledge processing at the sensor front-end**
- **Knowledge processing applications are often based on graph algorithms which in turn can be represented with sparse matrix algebra operations**
- **Sparse mapping framework allows for accurate modeling, representation, and mapping of fine-grained applications**
- **Initial results provide greater than an order of magnitude advantage over traditional 2D block cyclic distributions**

Acknowledgements

- MIT Lincoln Laboratory Grid (LLGrid) Team
- Robert Bond
- Pamela Evans
- Jeremy Kepner
- Zach Lemnios
- Dan Rabideau
- Ken Senne