



Automatic Mapping of the HPEC Challenge Benchmarks

Nadya T. Bliss
Jason Dahlstrom
Daniel Jennings
Sanjeev Mohindra

September 19th, 2006

MIT Lincoln Laboratory



Outline

- **Introduction**
- Automatic Mapping
- HPEC Challenge
- Results
- Summary

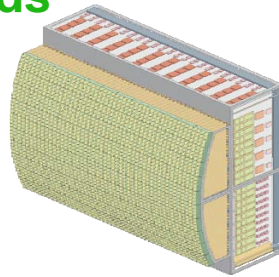


Next Generation Processing Trends

Array Technology Trends



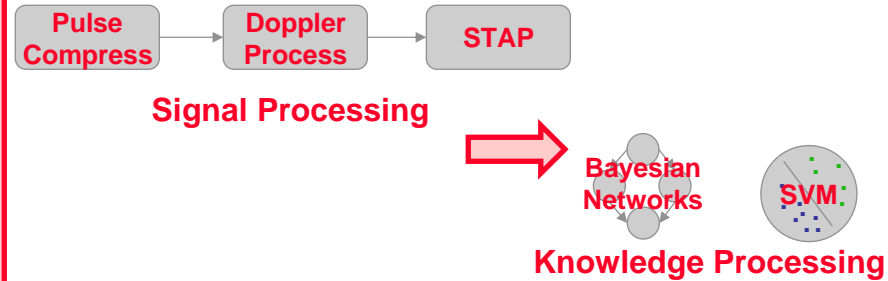
Analog array



Wideband digital array

Digital arrays require significantly more processing than arrays of the past

Algorithmic Trends

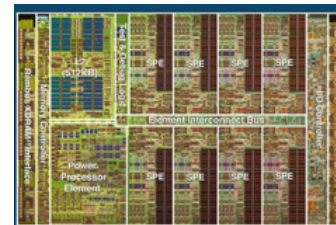


Applications require both signal and knowledge processing

Processor Technology Trends



Multi-processor system



Tile processor

To address the processing challenges parallel systems are being architected with multiple processing elements on a single chip.



Benchmark Motivation

Processor systems and architectures



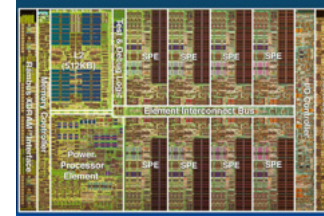
Single processor element



Cluster

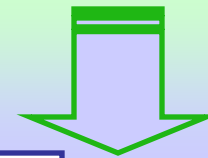


Multi-computer

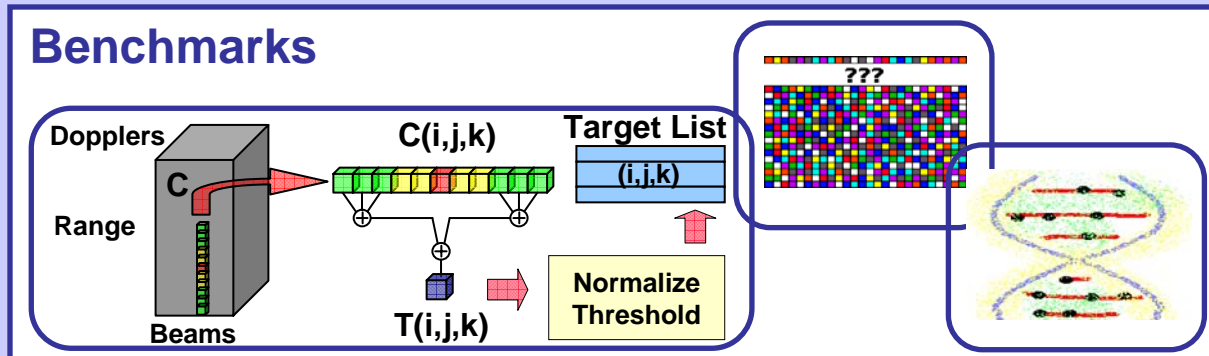


Tiled processor

Quantitative measures of performance are necessary to gauge the fitness of systems for emerging applications



Benchmarks

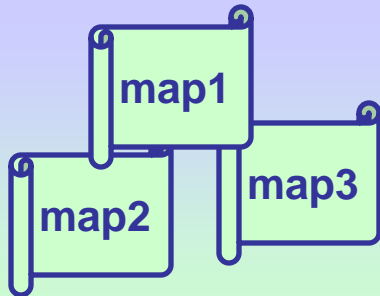
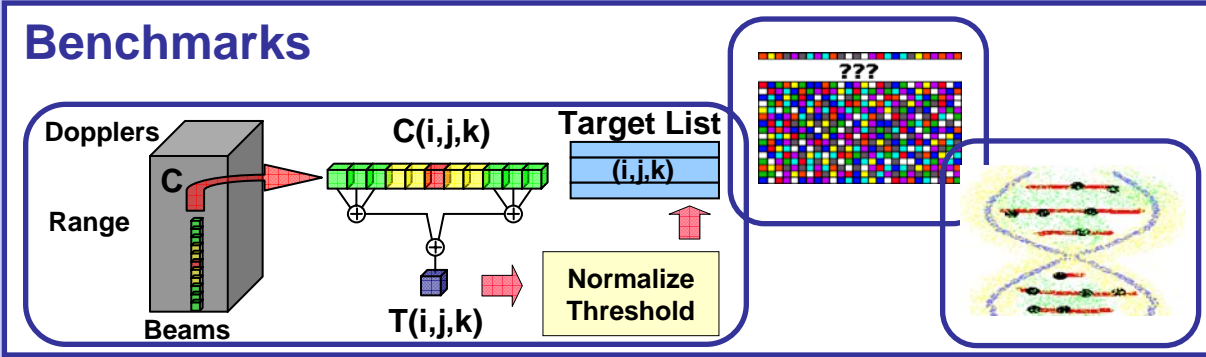


HPEC Challenge benchmarks allow for quantitative evaluation of processor systems and architectures.



Challenge: Mapping the Benchmarks

Benchmarks



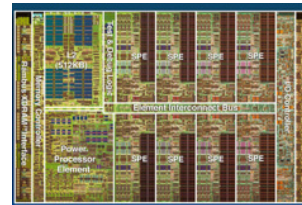
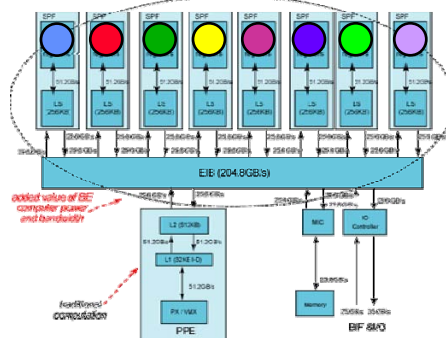
Mapping each benchmark is a major challenge

Use automatic mapping technology (pMapper) to both generate maps and predict architecture performance

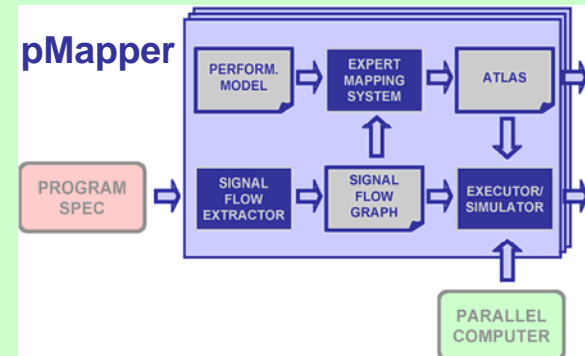
Processor

Cell BE

- 1 PPE
- 8 SPEs



pMapper





Outline

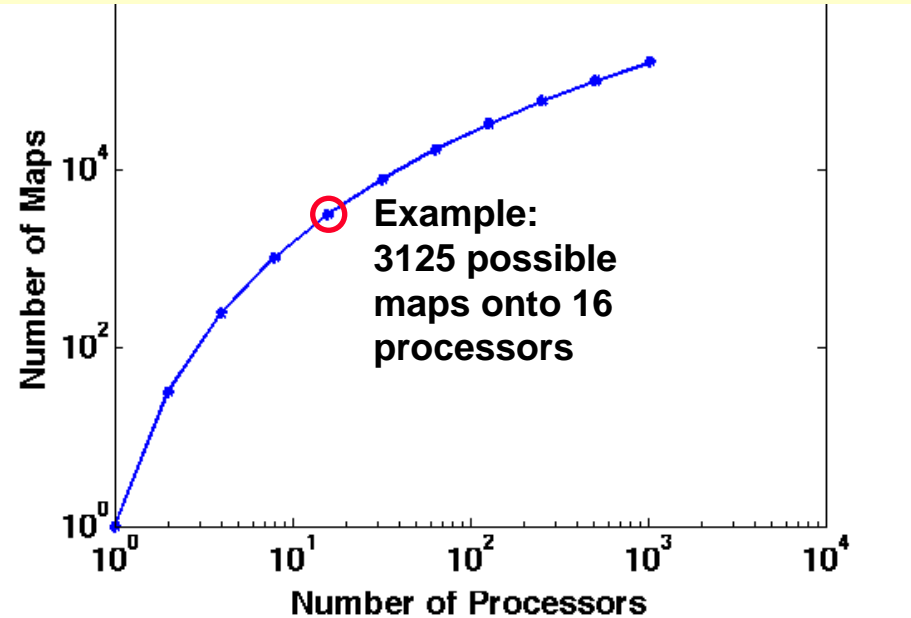
- Introduction
- **Automatic Mapping**
- HPEC Challenge
- Results
- Summary



Automatic Mapping: Why?

```
A = rand(N,M,mapA);  
B = zeros(N,M,mapB);  
C = zeros(N,M,mapC);  
D = rand(N,M,mapD);  
E = zeros(N,M,mapE);  
B(:, :) = fft(A, [], 1);  
C(:, :) = fft(B, [], 2);  
E(:, :) = C*D;
```

Number of possible maps for the sample program

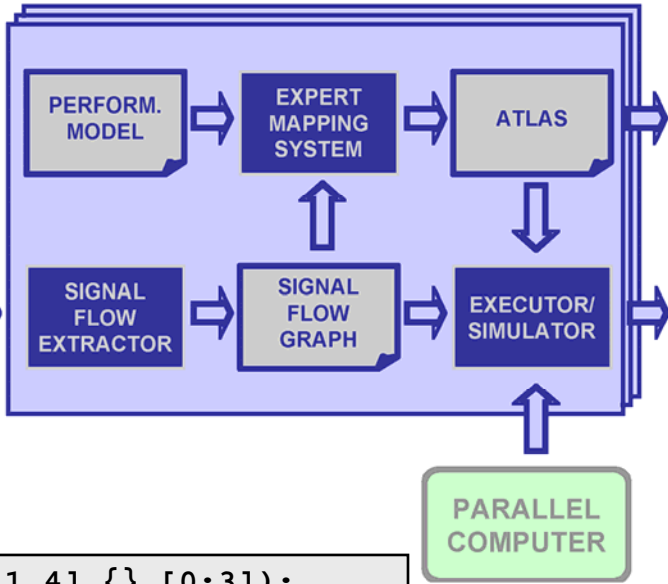


The size of the search space of possible maps for a computation is difficult (often impossible) to handle for a human



Automatic Mapping: pMapper

pMapper



pMapper is a middleware automatic mapping architecture that performs dynamic code analysis and global optimization.

```

mapA = map([1 4], {}, [0:3]);
mapB = map([4 1], {}, [0:3]);
mapC = map([2 2], {}, [0:3]);
mapD = map([2 2], {}, [0 2 1 3]);

A = rand(M,N,mapA);
B = zeros(M,N,mapB);
C = zeros(M,N,mapC);
D = rand(M,N,mapD);
E = zeros(M,N,mapD);

B(:, :) = fft(A, [], 1);
C(:, :) = fft(B, [], 2);
E(:, :) = D*C;
  
```

pMatlab

PARALLEL COMPUTER

maps are replaced with parallel tags

```

A = rand(M,N,p);
B = zeros(M,N,p);
C = zeros(M,N,p);
D = rand(M,N,p);
E = zeros(M,N,p);

B = fft(A, [], 1);
C = fft(B, [], 2);
E = D*C;
  
```

pMapper

Originally designed to map MATLAB programs to clusters.



Partial Maps

```

A = rand(M,N,p);
B = zeros(M,N,p);
C = zeros(M,N,p);
D = rand(M,N,p);
E = zeros(M,N,p);

B = fft(A,[],1);
C = fft(B,[],2);
E = D*C;

localeE = local(E);
localeE = foo(localeE);
E = put_local(E, localeE);

```

Initial pMapper design replaced maps with tags and used the performance model without restriction.

+
No mapping information has to be specified

-
Fragmented global array programming model inhibits optimization

Solution: allow for partial map specification and let pMapper fill in the missing information.

Full Map

```

Grid: 1x8
Dist: block
Procs: [0:7]

```

map attributes

Partial Map

```

Grid: 1x*
Dist:
Procs:

```

A partial map has one or more of the map attributes unspecified

```

A = rand(M,N,p);
B = zeros(M,N,p);
C = zeros(M,N,p);
D = rand(M,N,p);
pmap = map([1 *]);
E = zeros(M,N,pmap);

B = fft(A,[],1);
C = fft(B,[],2);
E = D*C;

localeE = local(E);
localeE = foo(localeE);
E = put_local(E, localeE);

```



Mapping Algorithm

- Use lazy evaluation to collect as much information as possible
- Store the program information as a signal flow graph
- Generate an *atlas* for the signal flow graph

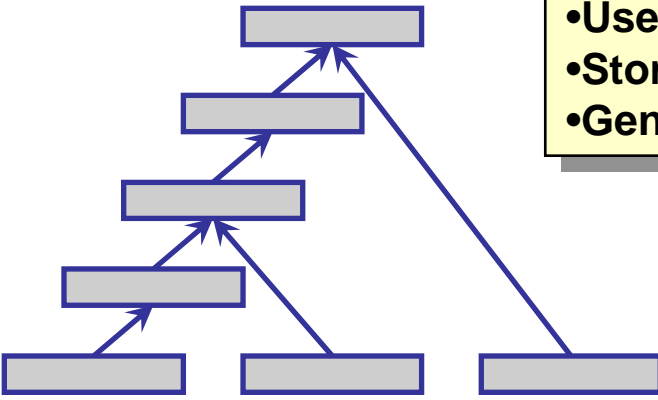


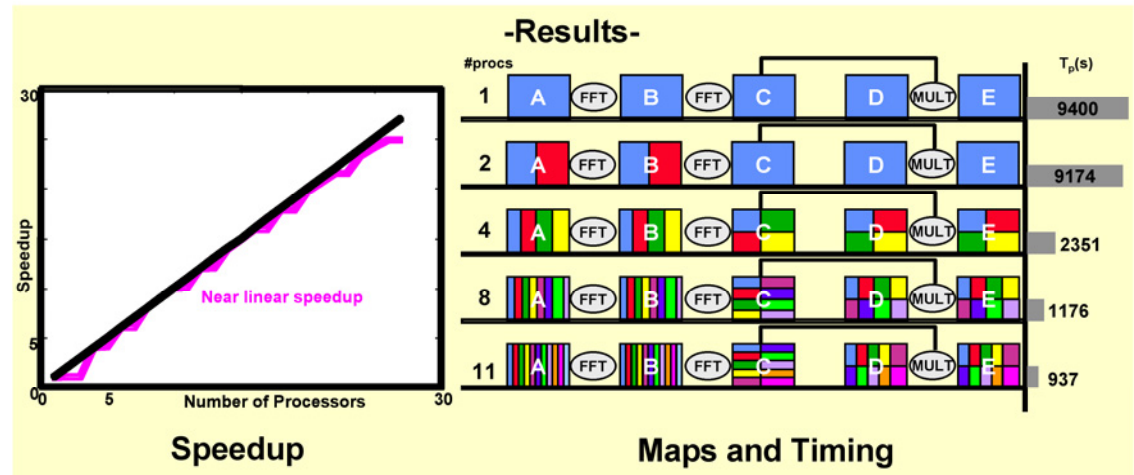
Table is built with an algorithm based on **dynamic programming**. Each new entry is generated based on previously generated entries.

	1	2	3	4	...
1					
2					
...					

Number of processors →

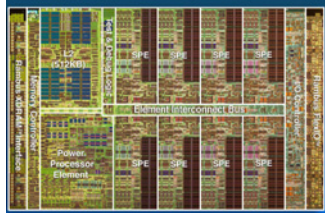
SFG Nodes ↓

Entry (i,j) contains the best atlas for the first i SFG nodes mapped on j processors.

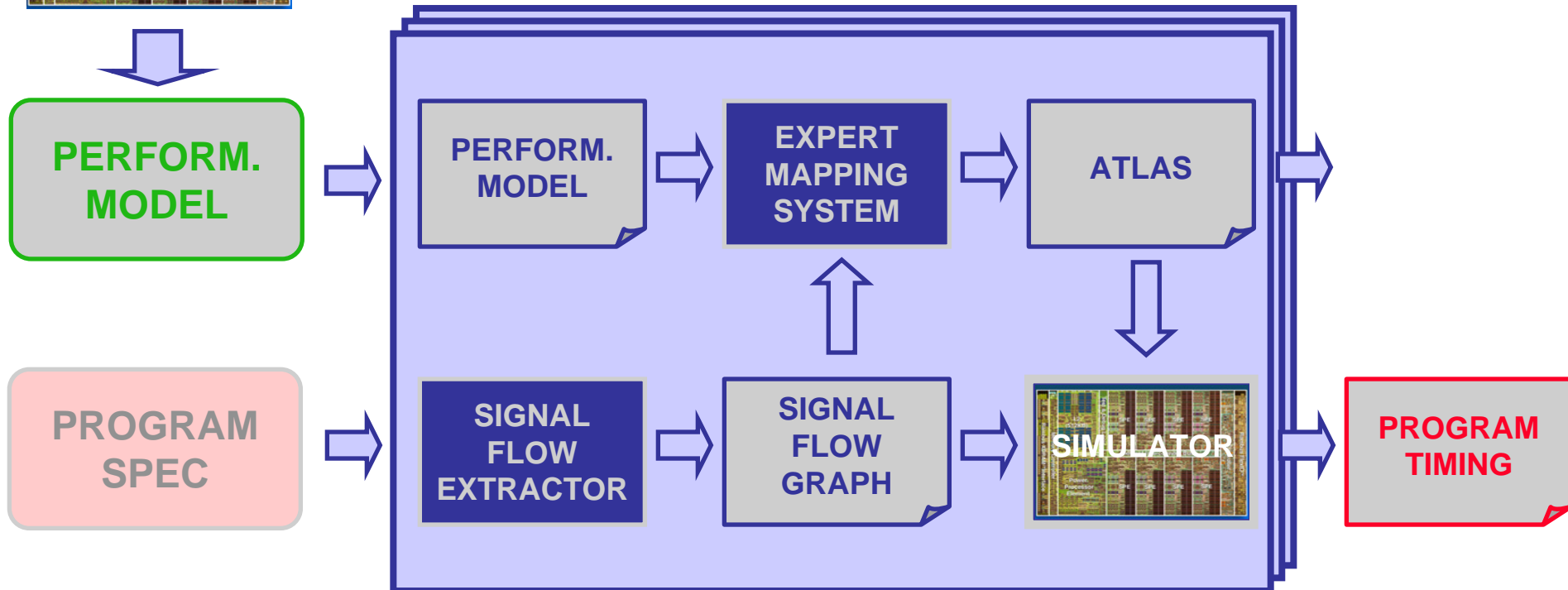




Simulated Mapping



pMapper can be used to generate maps using a program specification and a simulated system.

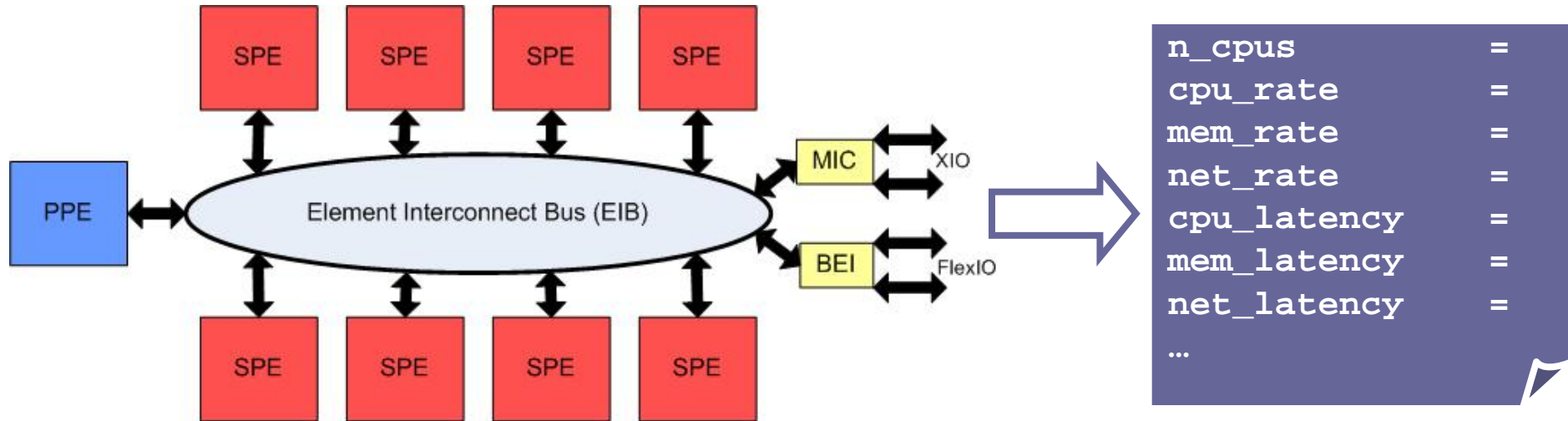


- Simulate the Cell BE processor using pMapper simulator infrastructure
- Use pMapper to predict mapping and performance on the Cell BE



Machine Model

Machine model provides description of underlying hardware.



IBM Cell characteristics*

- 8 SPEs (`n_cpus`)
- Peak FLOPS @ 3.2 Ghz: 204.8 GFLOPS (`cpu_rate`)
- Processor to Memory bandwidth: 25.6 GB/sec (`mem_rate`)
- Network bandwidth: 76.8 GB/sec (`net_rate`)
- ...



Outline

- Introduction
- Automatic Mapping
- **HPEC Challenge**
- Results
- Summary



HPEC Challenge Overview*

- DARPA PCA program kernel benchmarks
 - Single-processor operations
 - Drawn from many different DoD applications
 - Represent both “front-end” signal processing and “back-end” knowledge processing
- DARPA HPCS program Synthetic SAR benchmark
 - Multi-processor compact application
 - Representative of a real application workload
 - Designed to be easily scalable and verifiable

hpec challenge

[Home](#) | [Benchmarks](#) | [Software](#) | [Publications](#) | [Acknowledgments](#) | [Site Map](#) | [Login/Register](#)

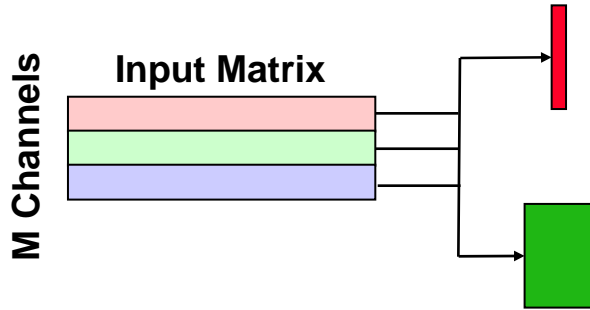
Introducing the HPEC Challenge Benchmark Suite

The embedded computing community is faced with an ever increasing challenge of producing software, firmware, and hardware to meet the demands of high performance commercial and DoD applications. These application requirements are driving the use of new computer processing elements with new processor architectures and increasing the complexity of application software.

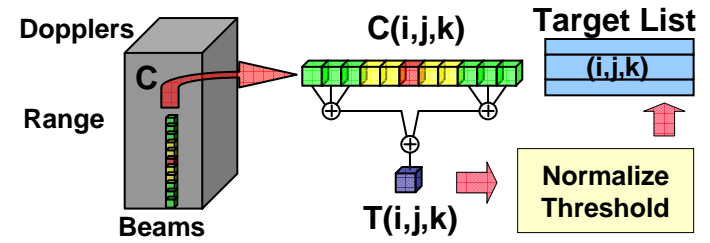


Signal Processing and Communication

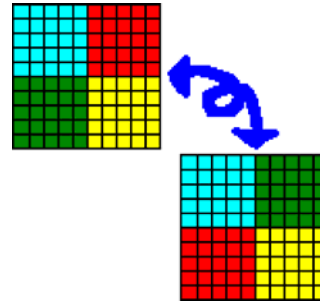
FIR



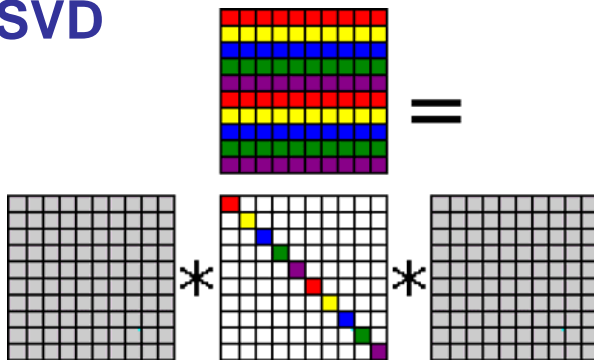
CFAR



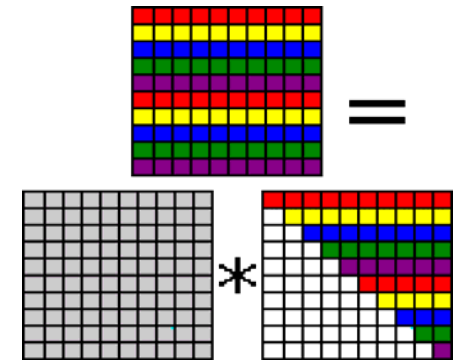
Corner Turn*



SVD



QR



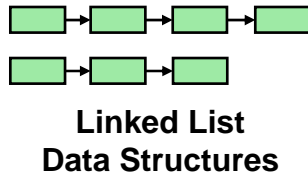
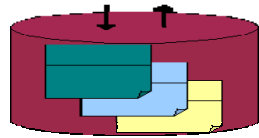
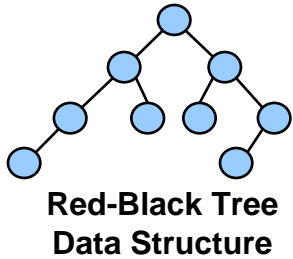
Front-end Processing

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

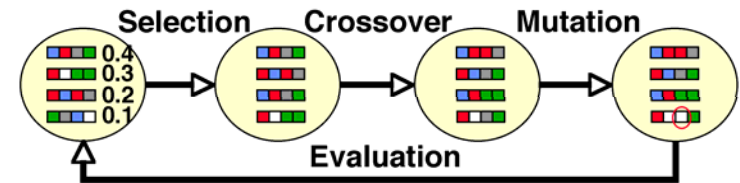
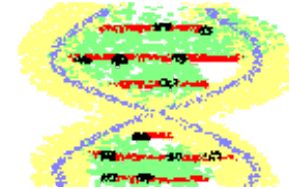


Knowledge Processing

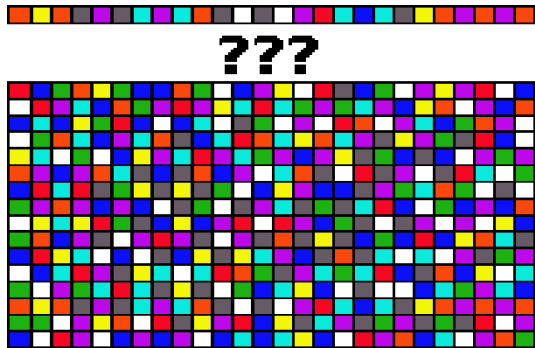
Database Operations



Genetic Algorithm



Pattern Match



- ### Back-end Processing
- Data dependent
 - Thread oriented
 - Information processing
 - Knowledge processing

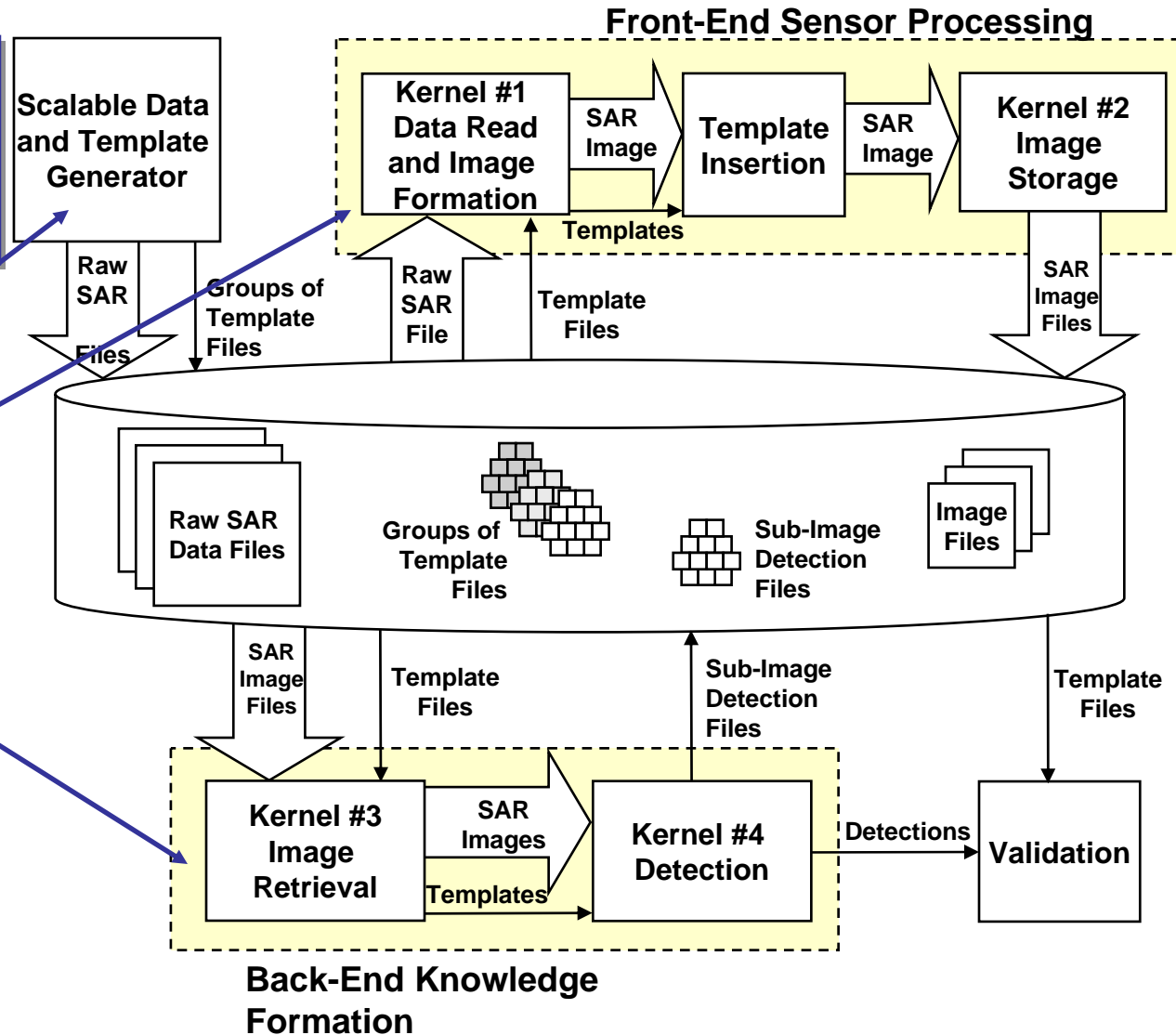


Application: Synthetic Aperture Radar

Intent of application benchmark

- Scalable
- High compute fidelity
- Low physical fidelity
- Self-verifying

Focus on computation





Outline

- Introduction
- Automatic Mapping
- HPEC Challenge
- **Results**
- Summary



Embarrassingly Parallel: FIR

Bank of input vectors



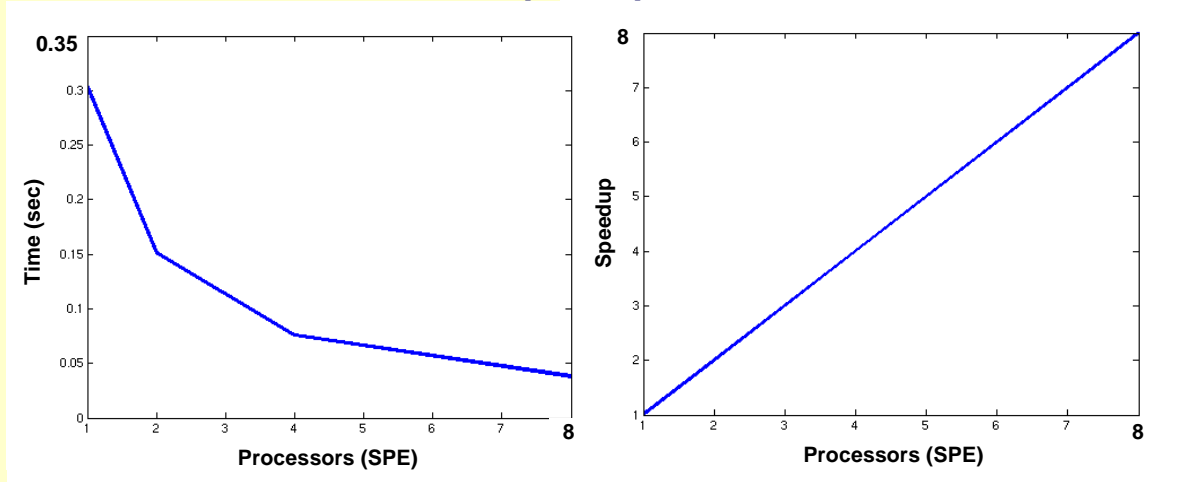
K Filter coefficients



Length of the input vector is equal to N and the number of input vectors is equal to M ,

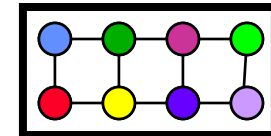
- The dataset can be represented as a $1 \times N \times M$ array
- The filter is small enough to be replicated

Time and Speedup for Dataset 2

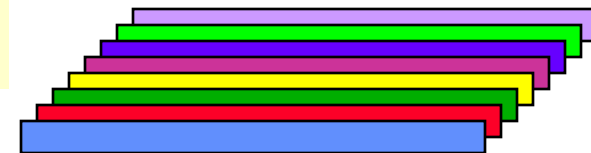


Dataset	N	M	K
1	4096	64	128
2	1024	20	12

Processors used:



Mapping:

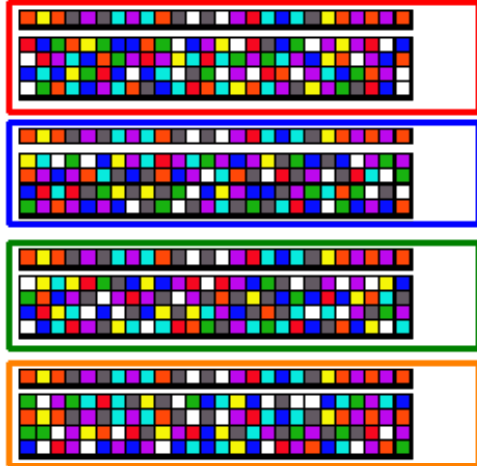


pMapper chooses the embarrassingly parallel mapping on 8 SPEs and produces *linear speedup*.



Embarrassingly Parallel

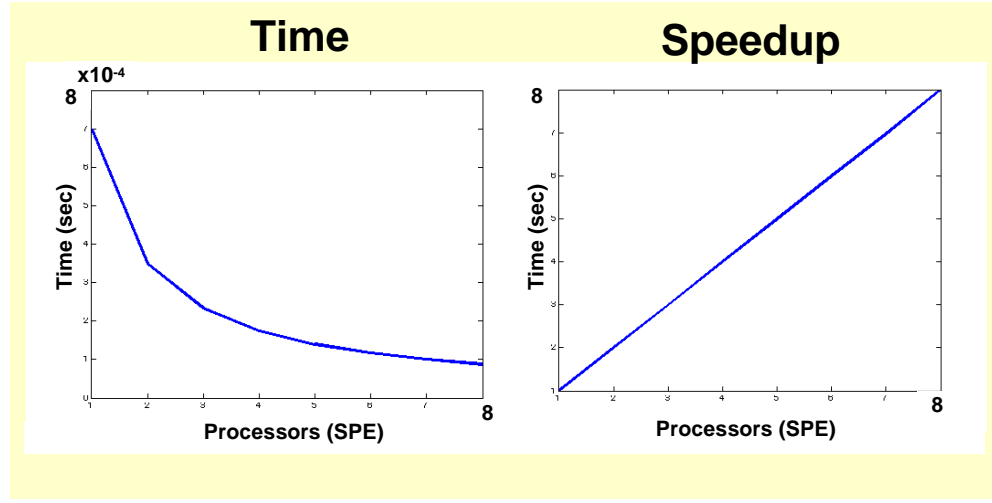
Pattern Match



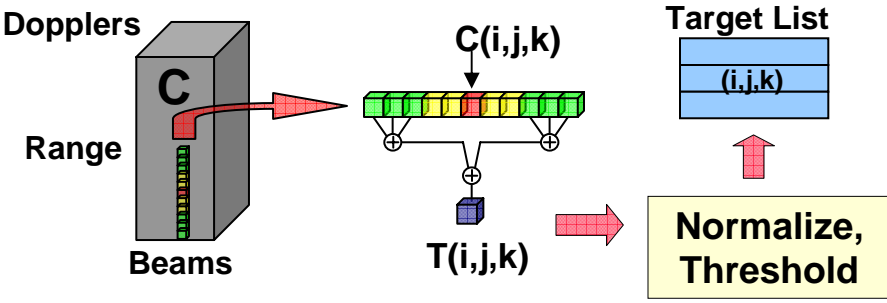
Mapping chosen:



Break the bank of patterns between processors

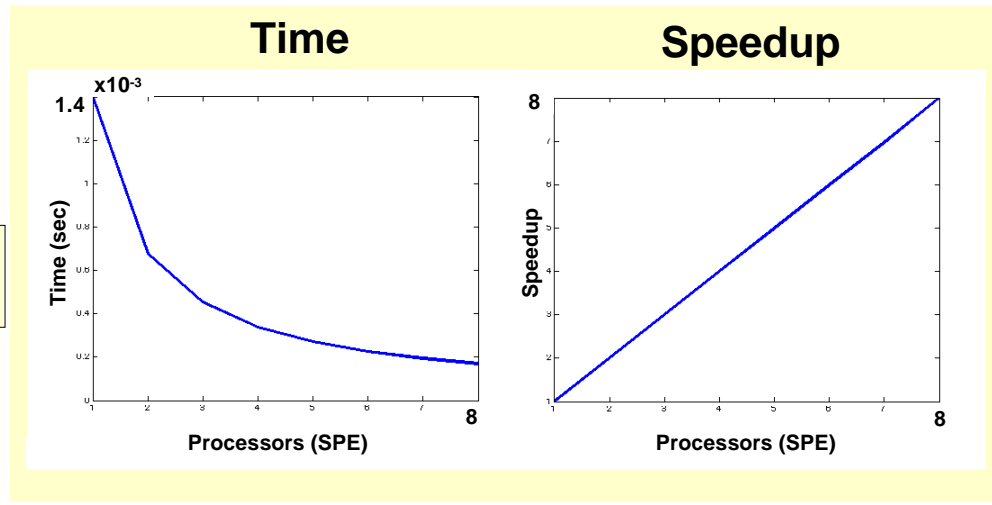


CFAR



Break up the data cube along the third (beams) dimension

Mapping chosen:





Genetic Algorithm

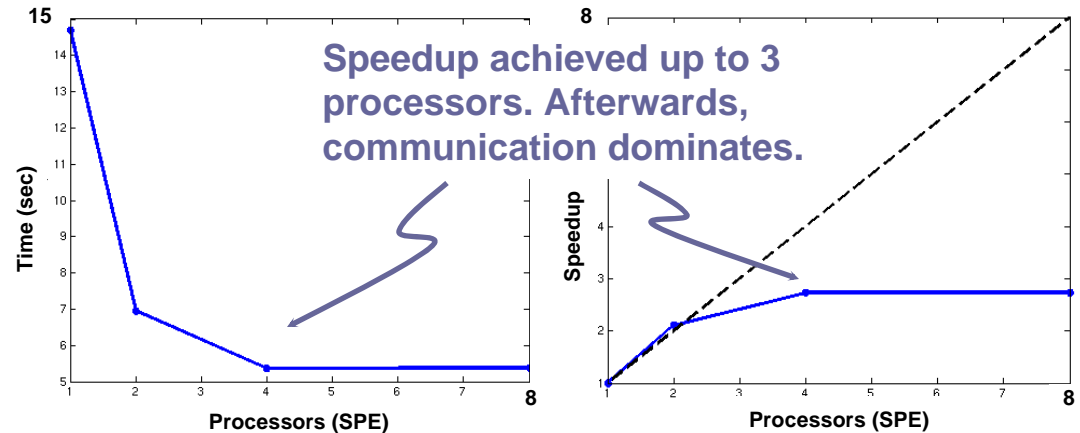
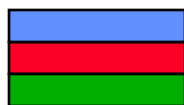
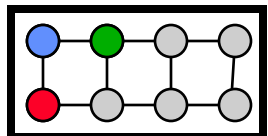
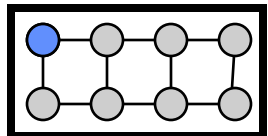
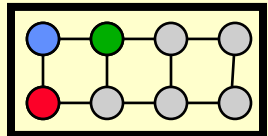
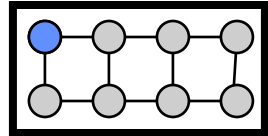
Dataset	N population	M chromosome
1	50	8
2	200	96
3	100	5
4	400	10

The algorithm implementation consists of

- Embarrassingly parallel crossover and mutation steps
- Communication step for each generation

Processors

Population



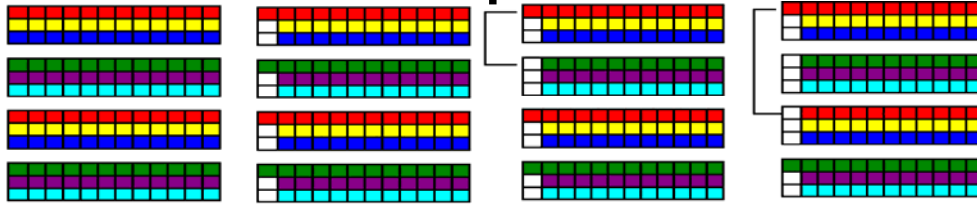
Observations:

- For smaller populations, communication step dominates
 - pMapper chooses to use a single processor (SPE)
- For larger populations, distribution is beneficial
 - pMapper correctly balances communication to computation ratio when determining the mapping

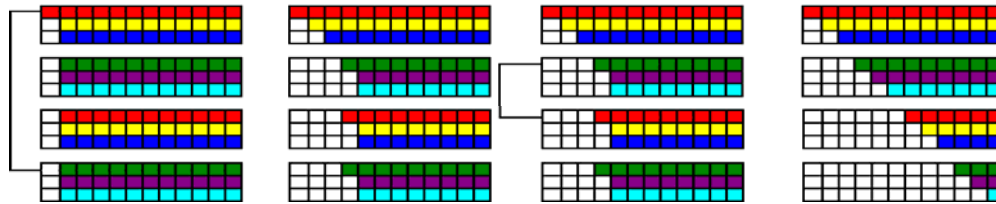


QR

Parallel Implementation



Global processing of Givens Rotation



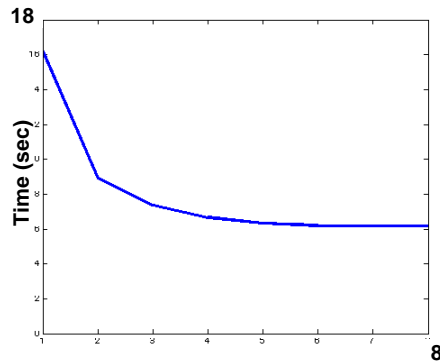
Mapping chosen:



At more than 6 processors, the communication starts to dominate.

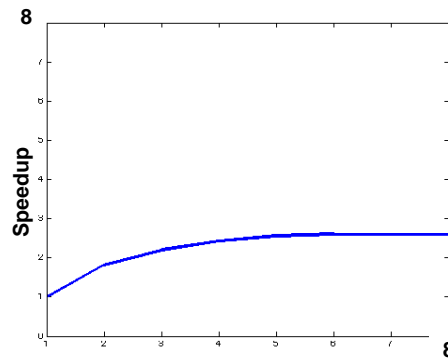
pMapper traverses the landscape of solutions and finds solutions that are in the valleys.

Time



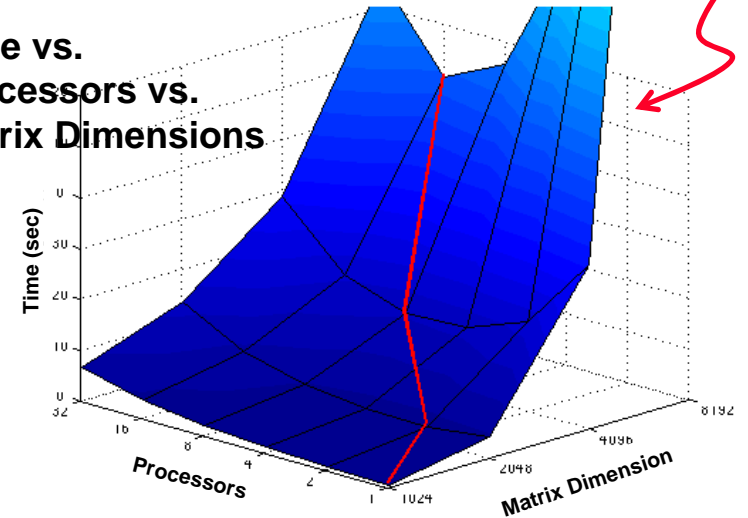
Processors (SPE)

Speedup



Processors (SPE)

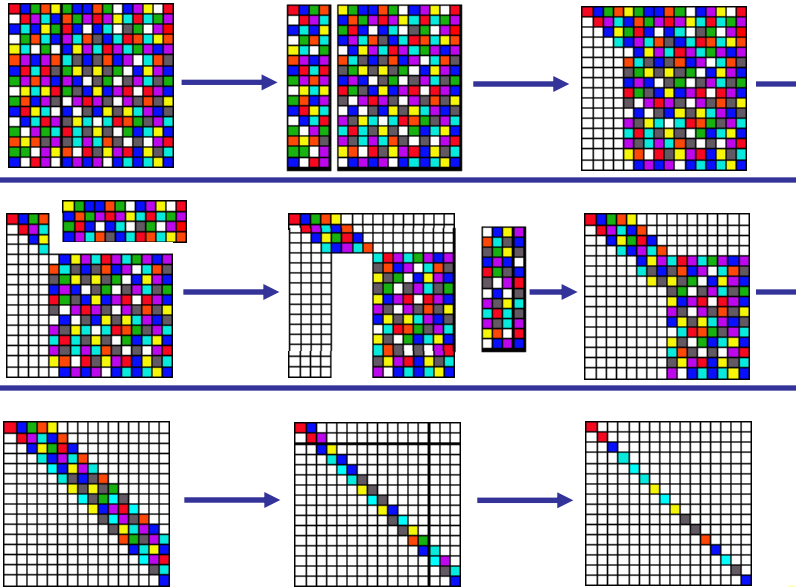
Time vs. Processors vs. Matrix Dimensions





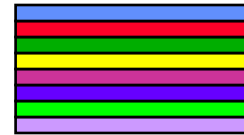
SVD

Parallel SVD via Block-Householder Bidiagonalization



The parallel SVD algorithm consists of both parallel and serial operations. The simulated results are provided for the bidiagonalization.

Mapping chosen:

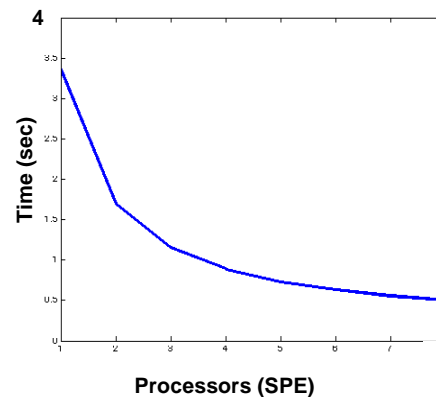


Past 8 processors, the communication starts to dominate.

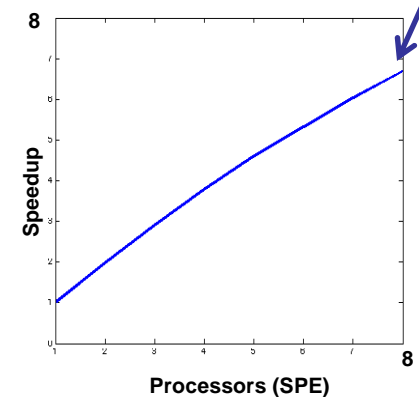
The algorithm requires a blocking variable h , which influences the timing data.

Input matrix 1024x1024 with $h = 16$.

Time



Speedup

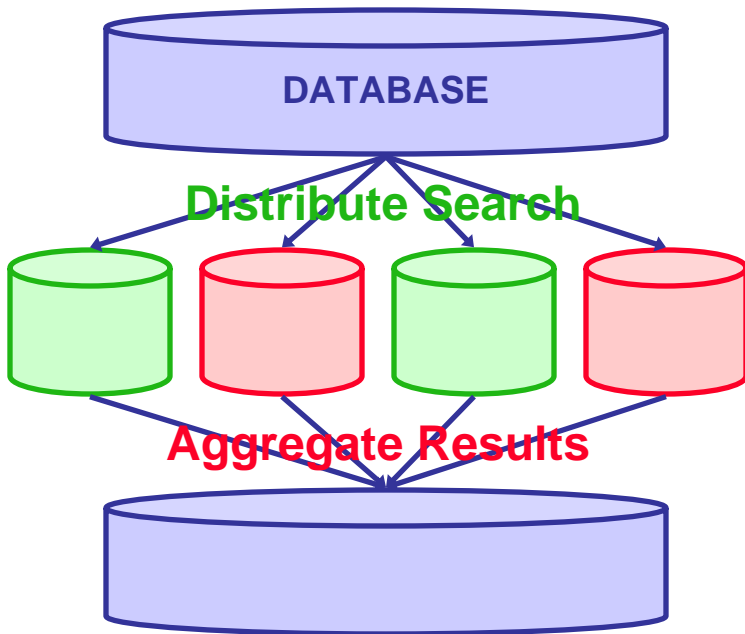




Database

Benchmark consists of database operations

- Insert - atomic, non parallelizable
- Delete - atomic, non parallelizable
- *Search - parallelizable*



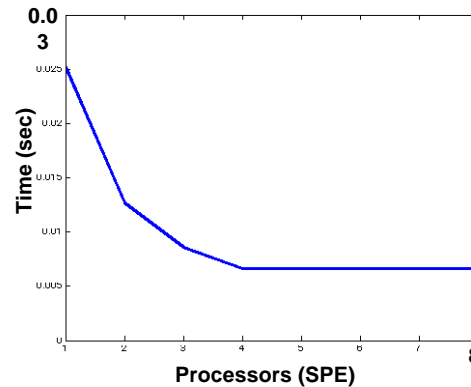
Communication operation starts to dominate after more than 4 processors.

Mapping chosen:

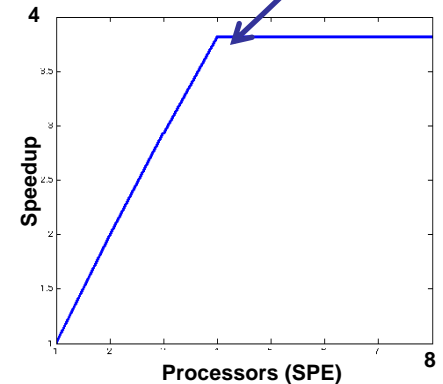


Cyclic mapping is chosen for the search operation - better load balancing than block distribution.

Time



Speedup

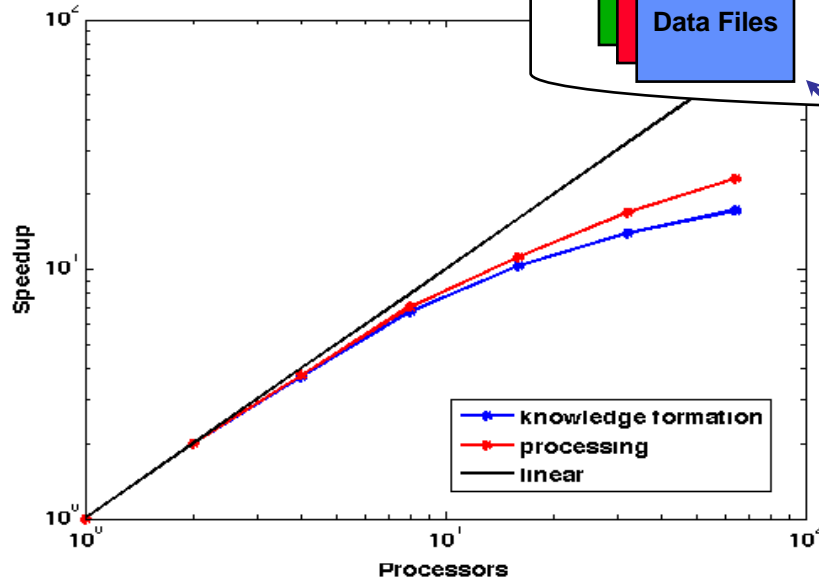
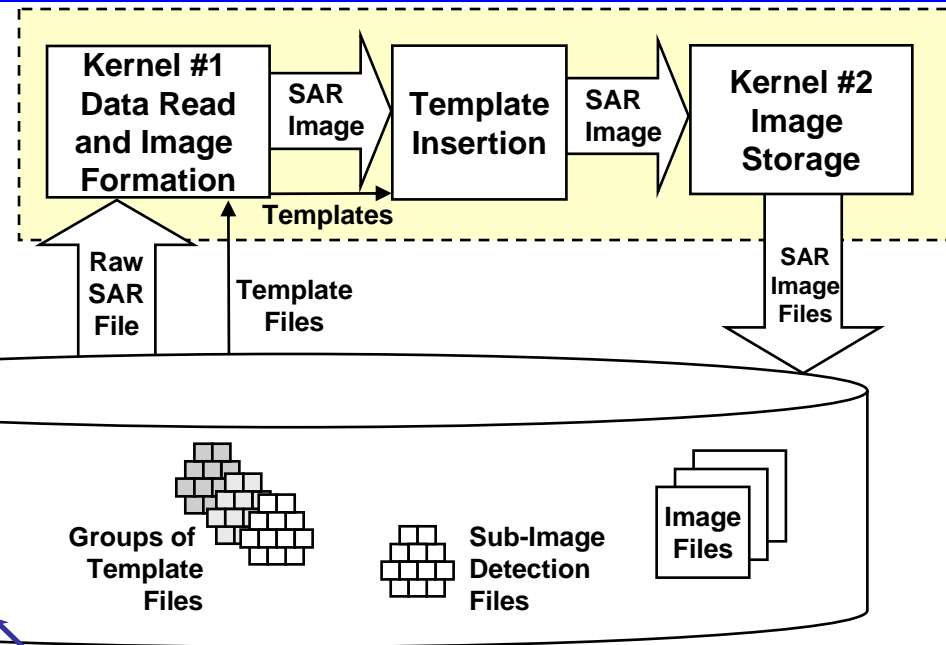




Application

Approach

- Replace maps with partial maps
- Use pMapper to run and execute on LLGrid



Results

- Linear speedup
- Embarrassingly parallel mapping



Outline

- Introduction
- Automatic Mapping
- HPEC Challenge
- Results
- **Summary**



Summary

Benchmark	Map	Speedup (max=8*)
FIR		8
CFAR		8
SVD		6.7
QR		2.6
Pattern Match		8
Genetic Algorithm		2.8
Database Operations		3.8
Application (SAR)		17/23**

pMapper finds efficient mappings for all of the benchmarks and is sensitive to algorithm parameters.



Acknowledgements

- **Robert Bond**
- **Ryan Haney**
- **Jeremy Kepner**
- **Hahn Kim**
- **Daniel Kunkle**
- **Julia Mullen**
- **Edward Rutledge**
- **Sharon Sacco**
- **Glenn Schrader**
- **Ken Senne**