pMapper:

Automatic Mapping of Parallel MATLAB Programs*

Nadya Travinin
Henry Hoffmann
Robert Bond
Hector Chan
Jeremy Kepner
Edmund Wong

September 21th, 2005

*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.

Acknowledgements

Daniel Jennings

Hahn Kim

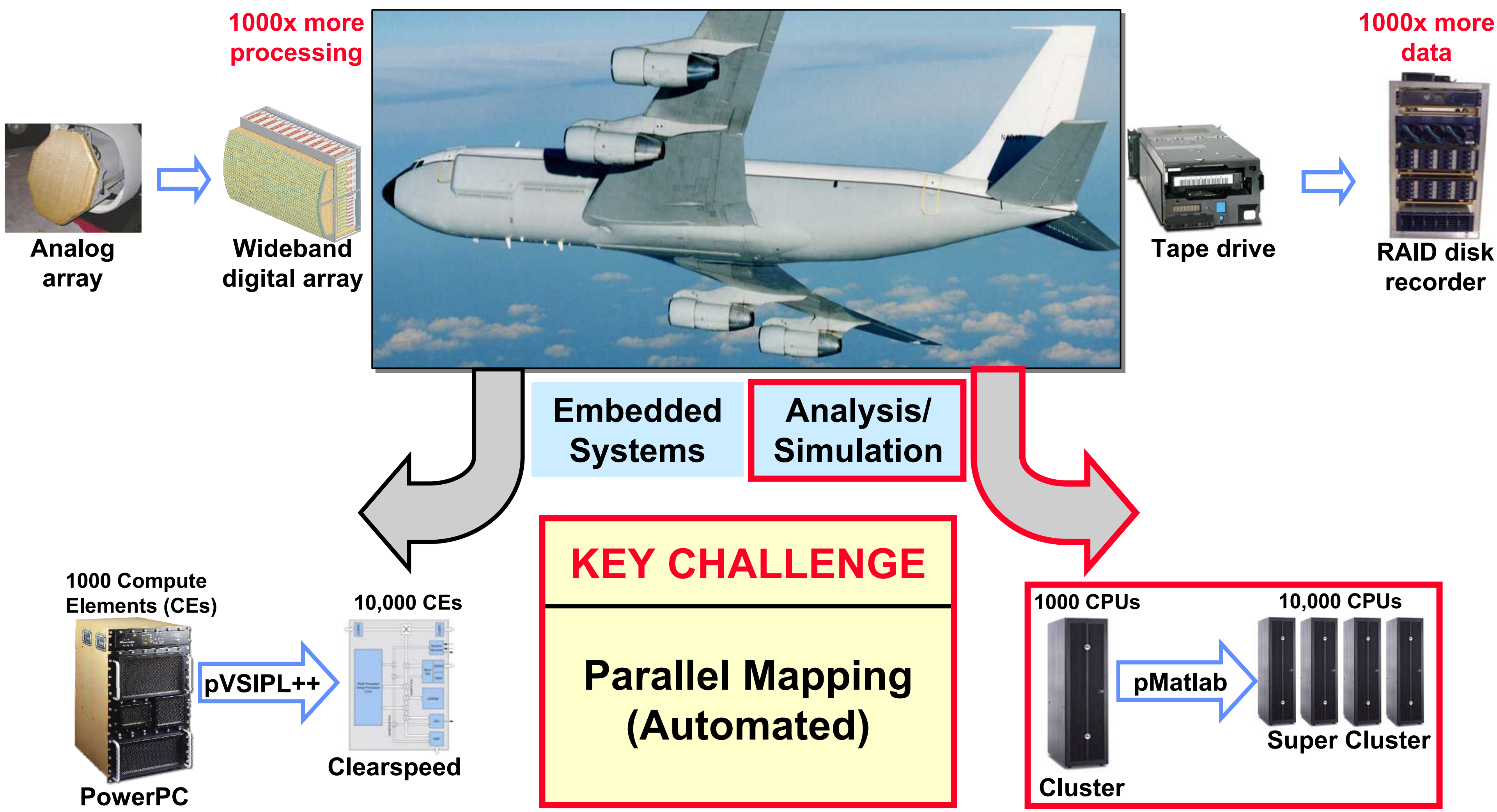
Jeff Lebak

Albert Reuther

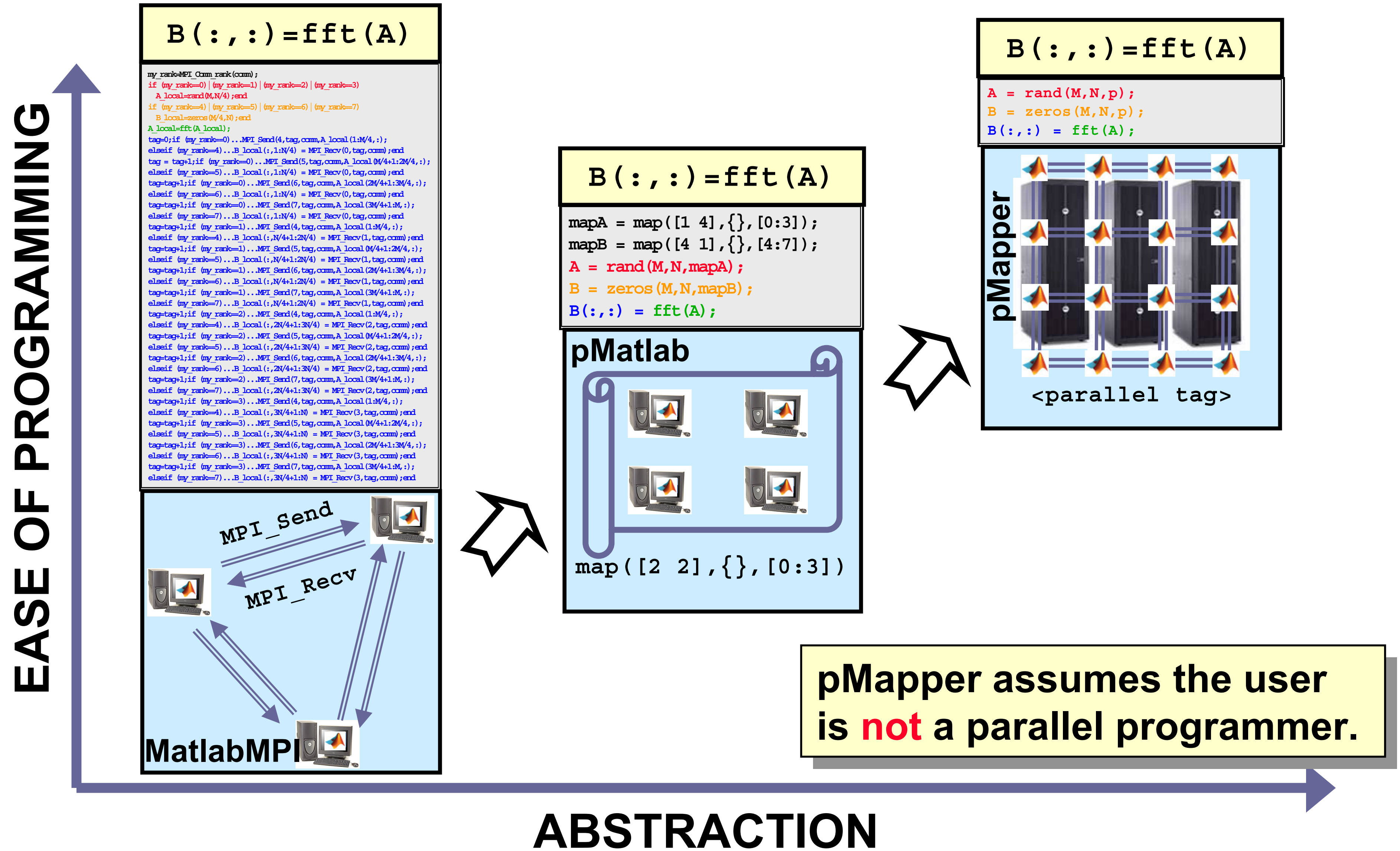
Outline

- **Introduction**
- Automated Parallel Mapping
- Preliminary Results
- Summary

Next Generation
Sensor and Image Processing

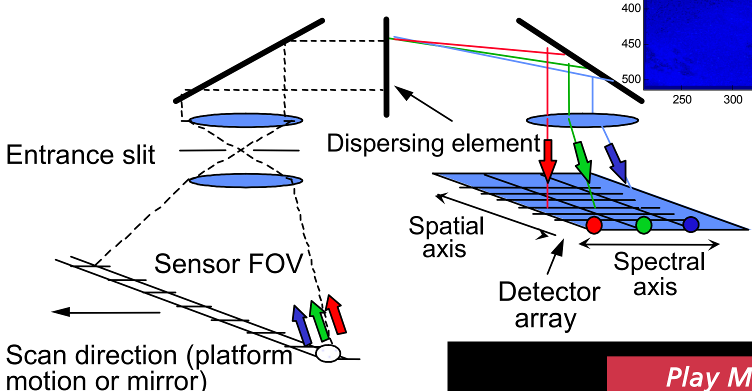


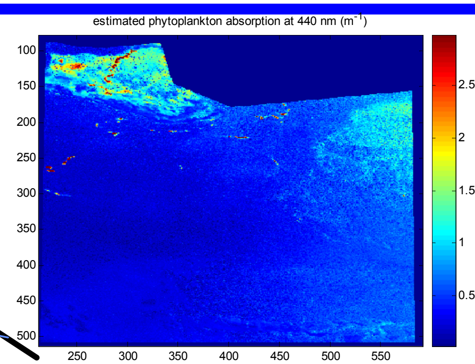
Evolution of Parallel Programming



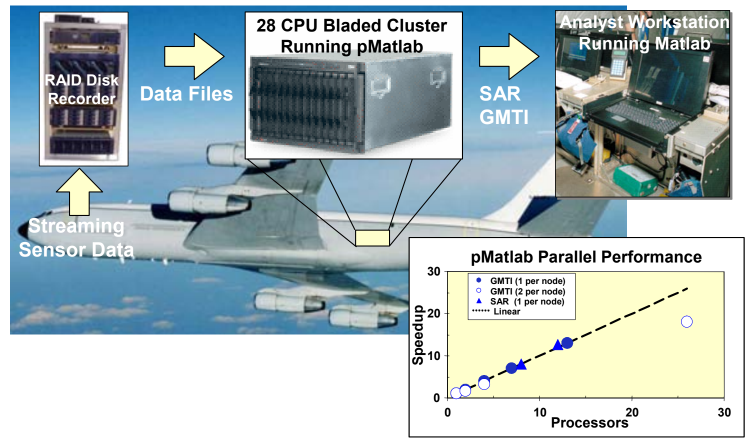
pMatlab Applications

Hyperspectral imaging,
David Stein

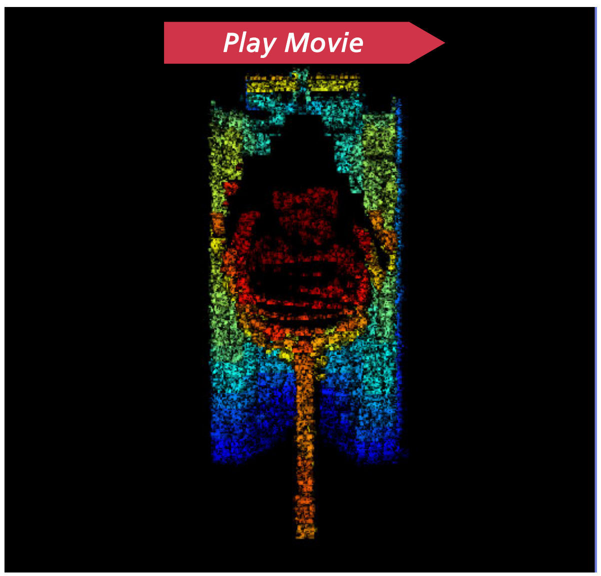




"QuickLook" SAR and GMTI processing*



Parallel Coherent Ladar Simulator,
Mark Rubin



pMatlab has been used widely throughout the Lincoln Laboratory from optical SAR simulations to quicklook processing of data in flight.

*J. Kepner, T. Currie, H. Kim, A. McCabe, B. Mathew, M. Moore, D. Rabinkin, A. Reuther, A. Rhoades, N. Travinin, and L. Tella, "Deployment of SAR and GMTI Signal Processing on a Boeing 707 Aircraft using pMatlab and a Bladed Linux Cluster," *HPEC Workshop 2004*, Lexington, MA, September 2004.

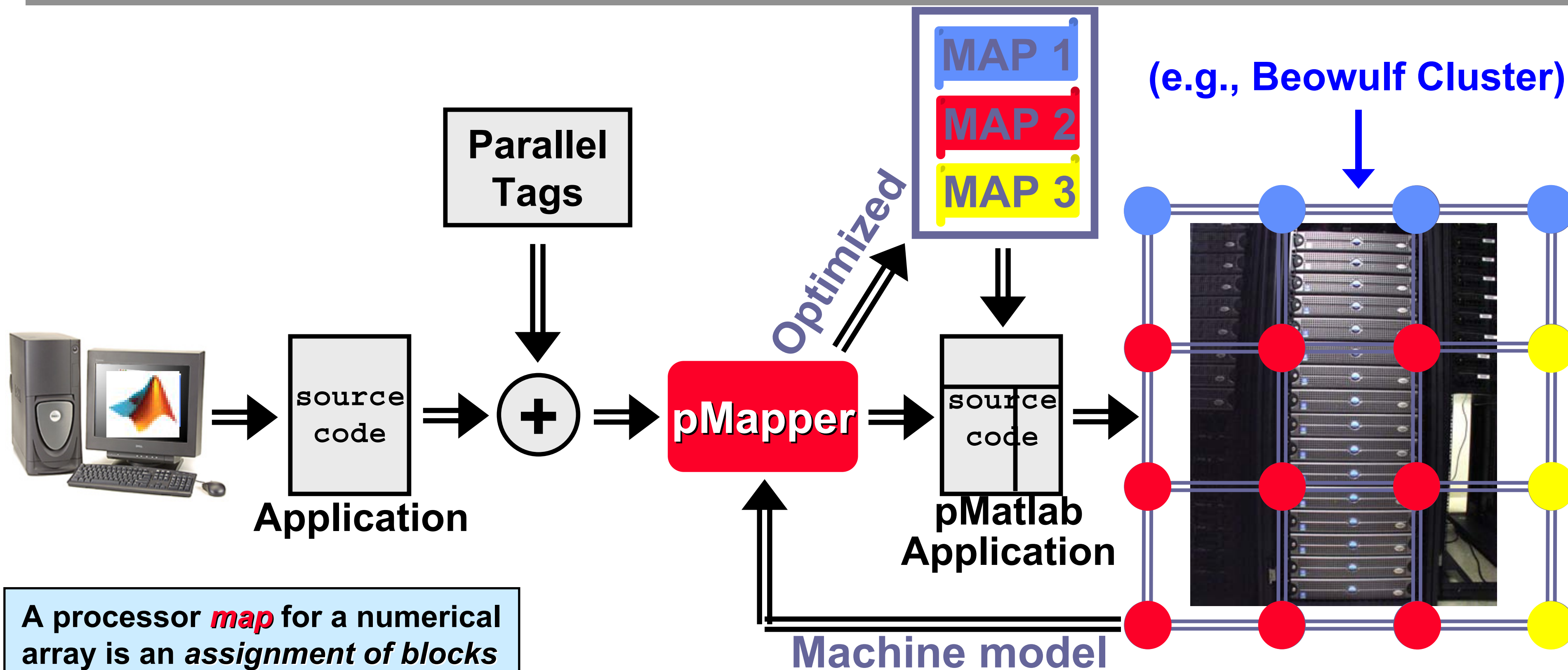
Outline

- Introduction
- **Automated Parallel Mapping**
  - **Motivation and Goals**
  - **Architecture**
- Preliminary Results
- Summary

pMapper Overview

System for running large signal processing applications on parallel machines, satisfying two sub goals:

Faster time to solution (optimized mapping)
Ease of programming (automated mapping)



A processor *map* for a numerical array is an *assignment of blocks of data to processing elements*.

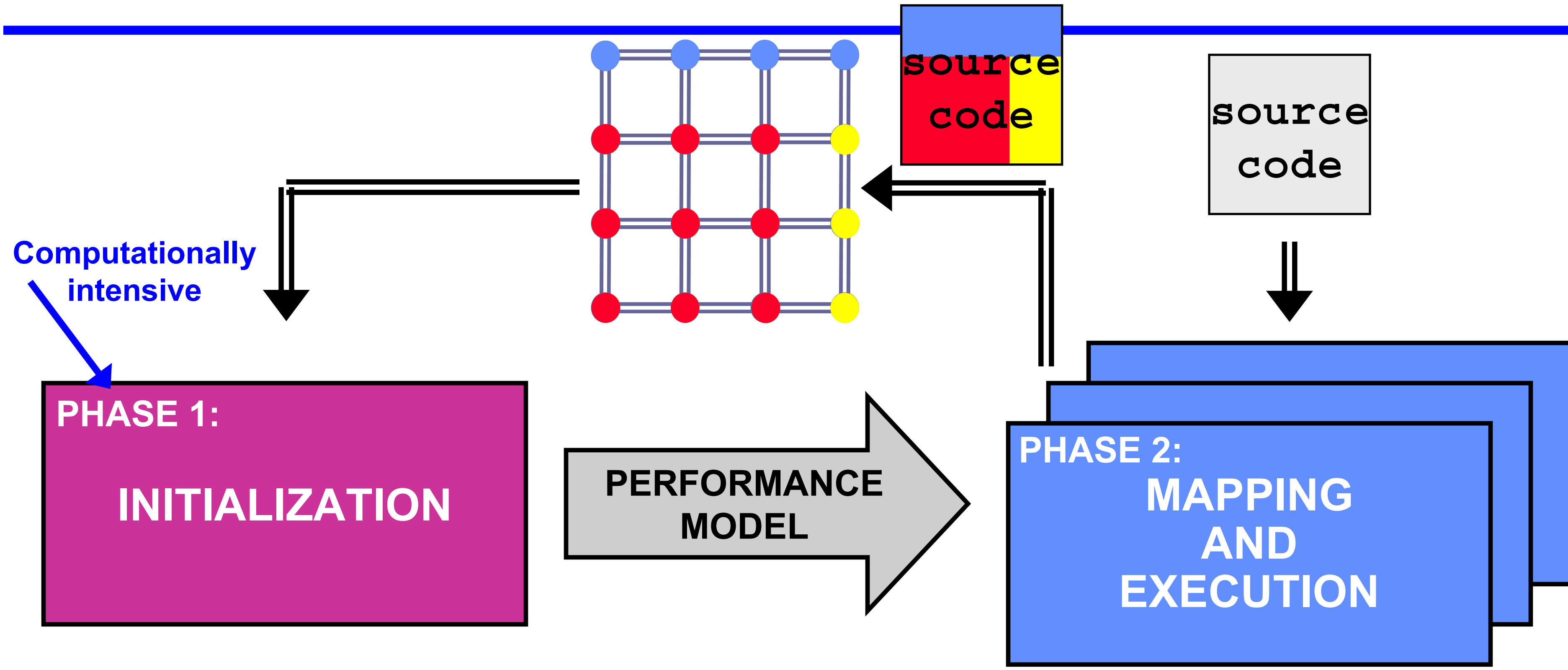
Taxonomy of Automated Mapping Approaches

CONCURRENCY	Serial	Parallel	pMapper
SUPPORT LAYER	Compiler	Middleware	
CODE ANALYSIS	Static	Dynamic	
OPTIMIZATION WINDOW	Local/ Peephole	Global/ Program Flow	

Examples

1. FFTW: serial & parallel, compiler, static, local
2. Streamit: parallel, compiler, static, global&local
3. pH: parallel, compiler, static, local
4. ATLAS: serial, middleware, static, local
5. Dynamo: serial, compiler, dynamic, local
6. pMapper: parallel, middleware, dynamic, global

2 Phase Mapping Architecture



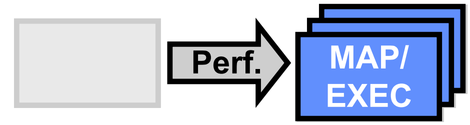
- **Done once** for a given parallel computer and parallel MATLAB library version
- **Generates the performance model** used in mapping and execution phase

- **Done for every** parallel MATLAB program
- **Uses the performance model** in making parallel mapping decisions

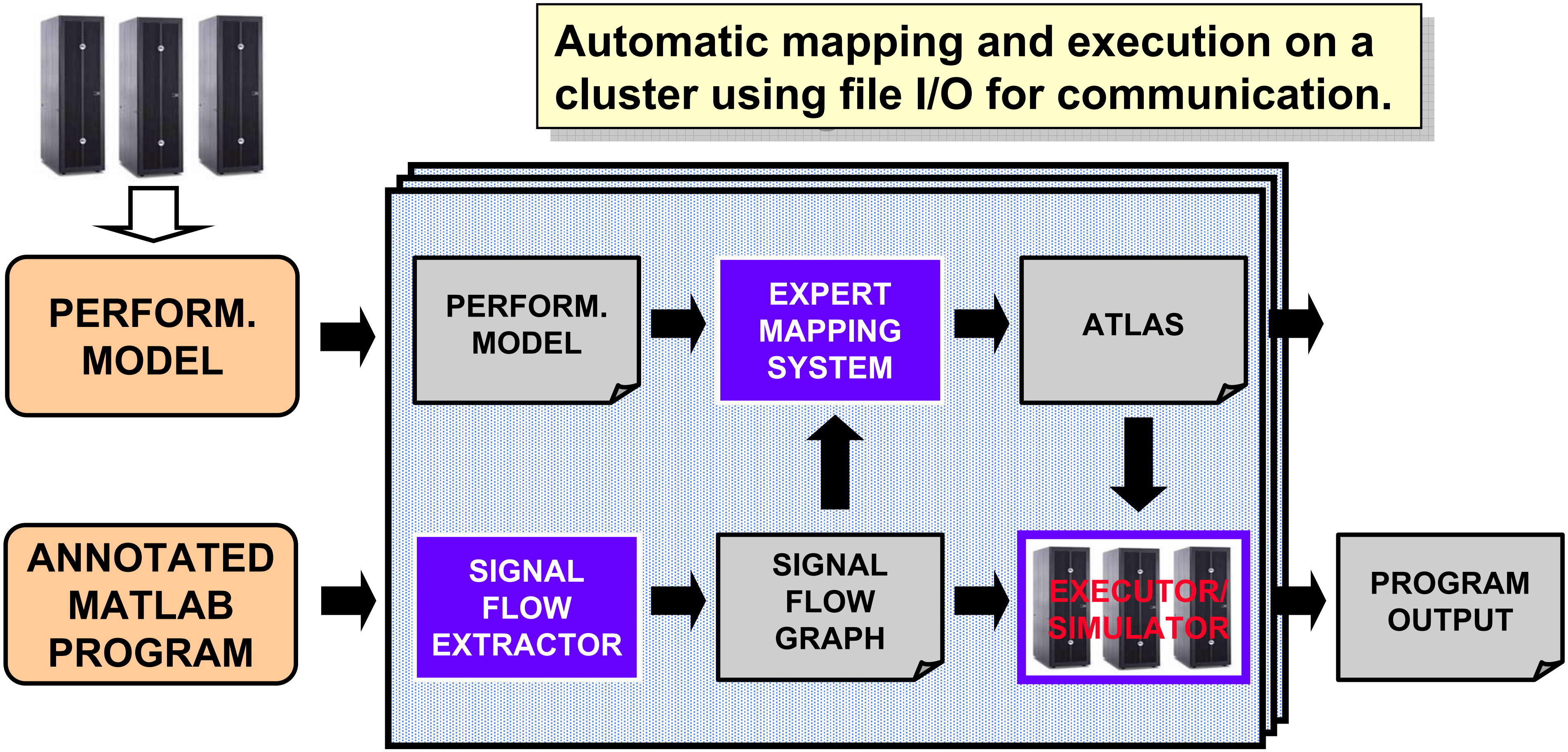
Architecture supports multiple mapping approaches.

Mapping and Execution

Phase 2



Automatic mapping and execution on a cluster using file I/O for communication.

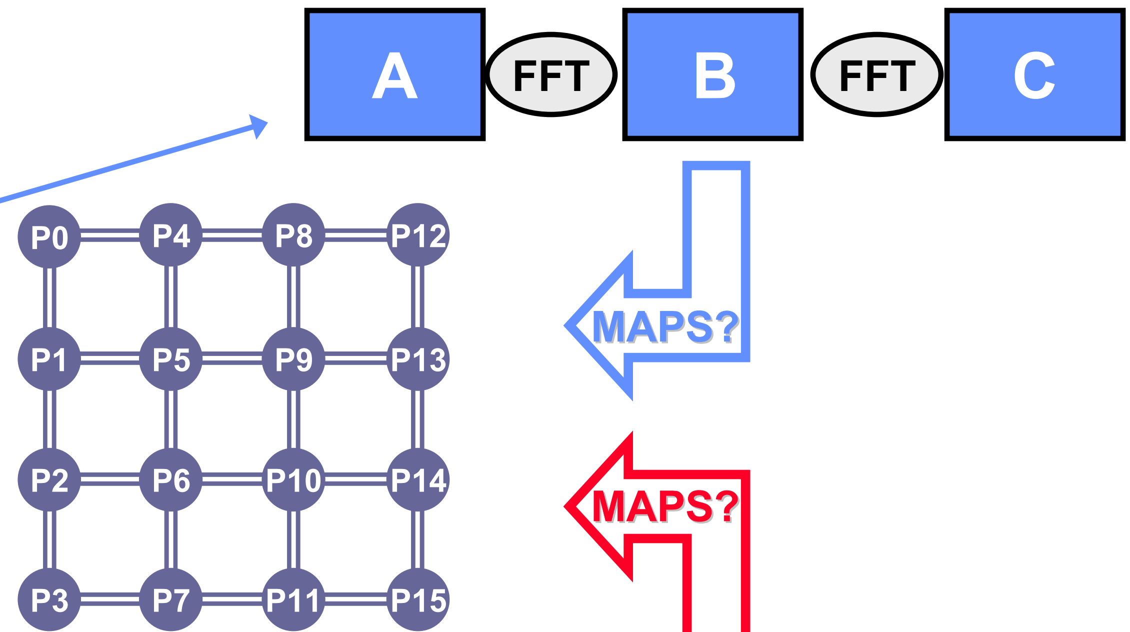


Signal Processing Mapping Challenges

Multi-stage Application

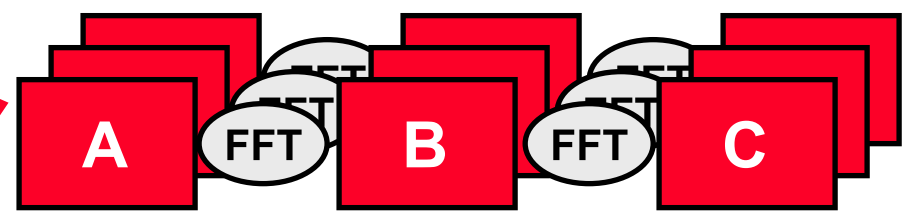
```
B(:,:) = fft(A,[],1);
C(:,:) = fft(B,[],2);
C
```

Generates a chain signal flow graph



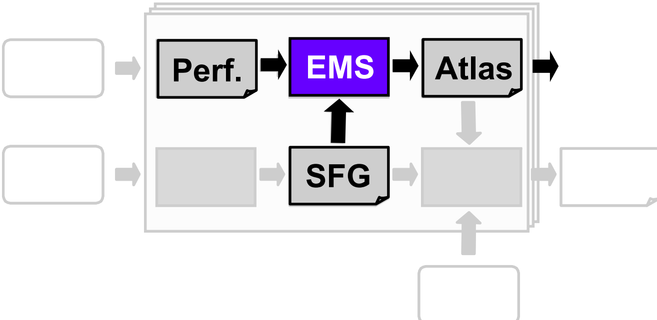
Multi-pipeline Application

```
for i = 1:M
  B(:,:,i) = fft(A(:,:,i),[],1);
  C(:,:,i) = fft(B(:,:,i),[],2);
end
C
```

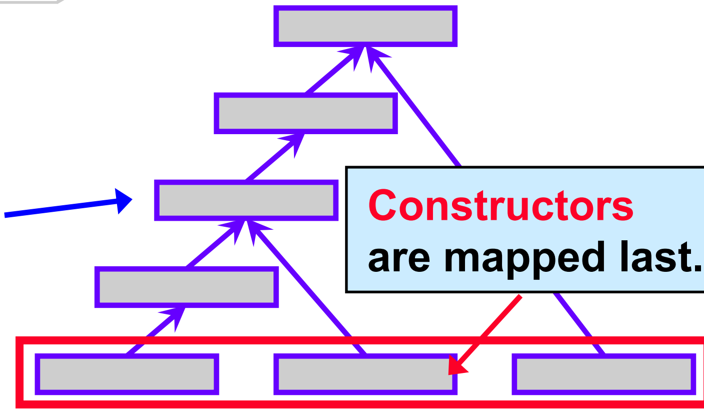


Generates a tree signal flow graph, adding an extra level of complexity.

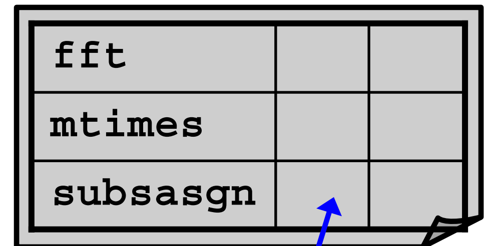
Multi-stage Mapping Algorithm



The **Expert Mapping System** produces an **atlas** for the signal flow graph.



Constructors are mapped last.



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

Quality of each atlas is determined by the performance model.

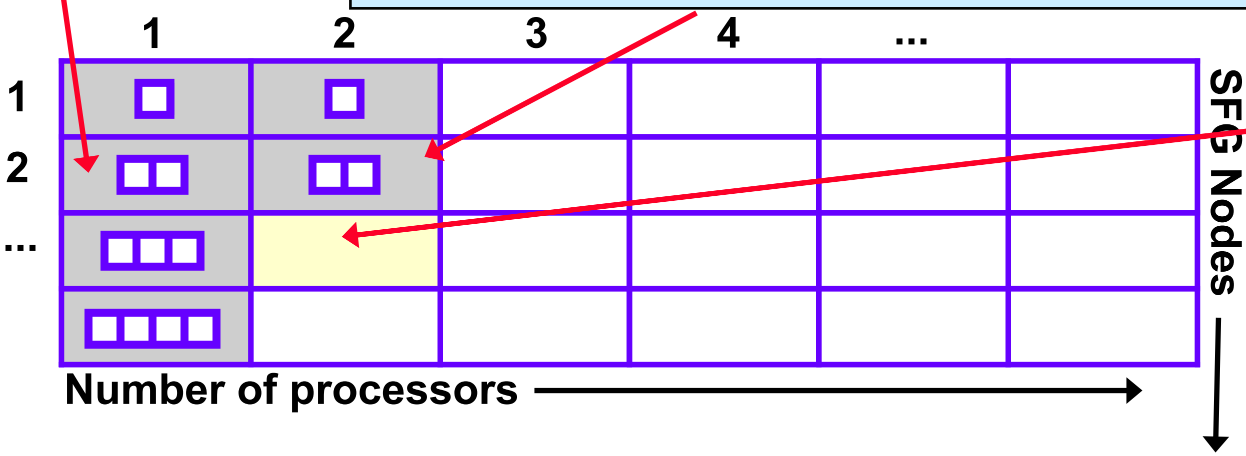
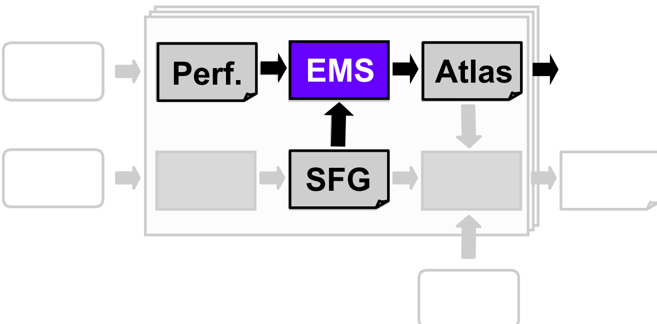
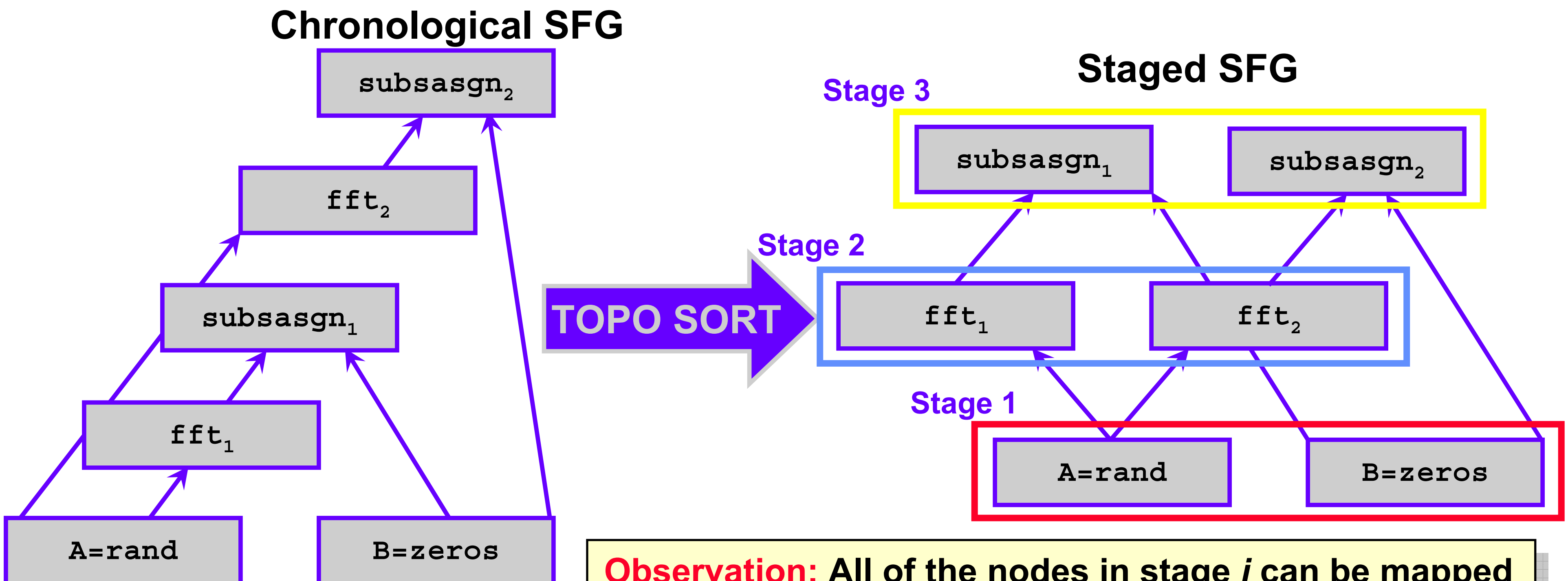


Table is built with an algorithm based on **dynamic programming**. Each **new** entry is generated **based on previously** generated entries.

Multi-pipeline Mapping Algorithm



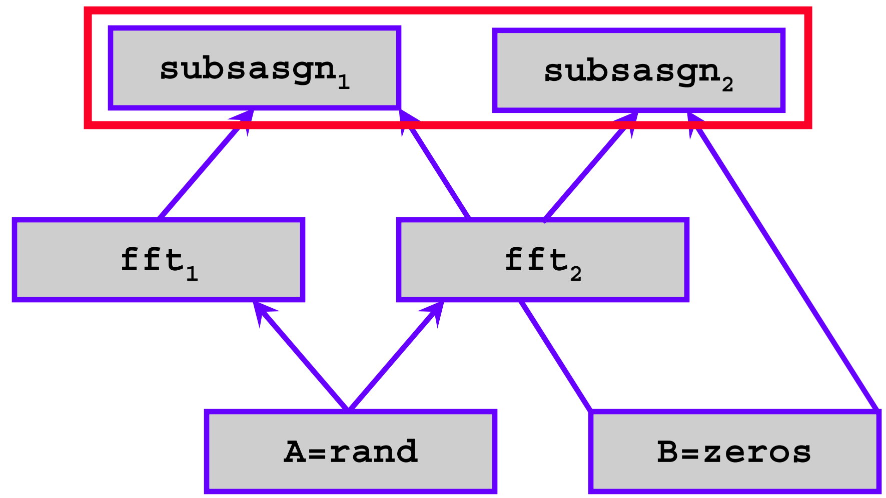
Step 1: Arrange the nodes of the signal flow graph into stages of computation by performing a **topological sort**.

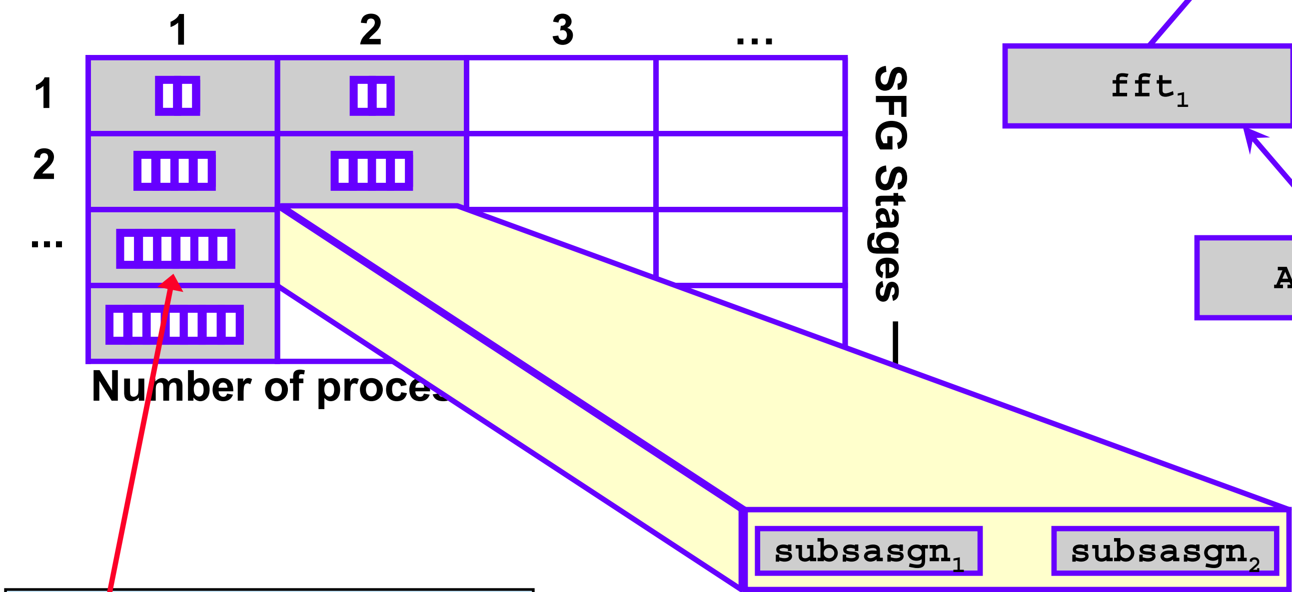


Observation: All of the nodes in stage *i* can be mapped independently of all the other nodes in stage *i*.

Mapping Algorithm (Cont.)

Step 2: Map each stage by assigning the number and ranks of processors.





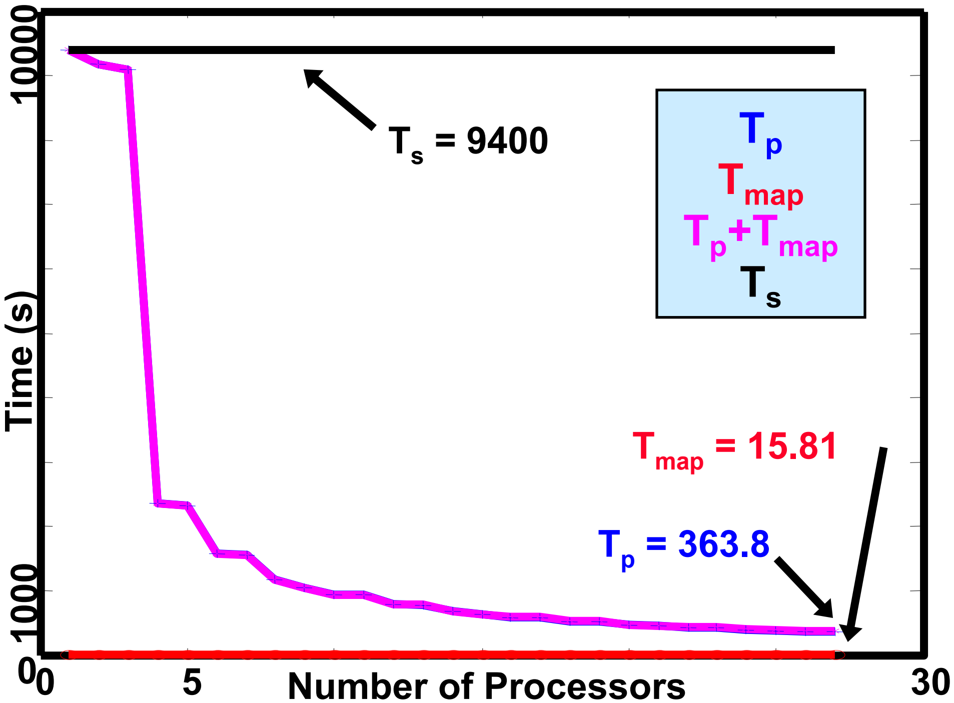
Cell (i,j) contains the best atlas for the first i **SFG stages** mapped on j processors.

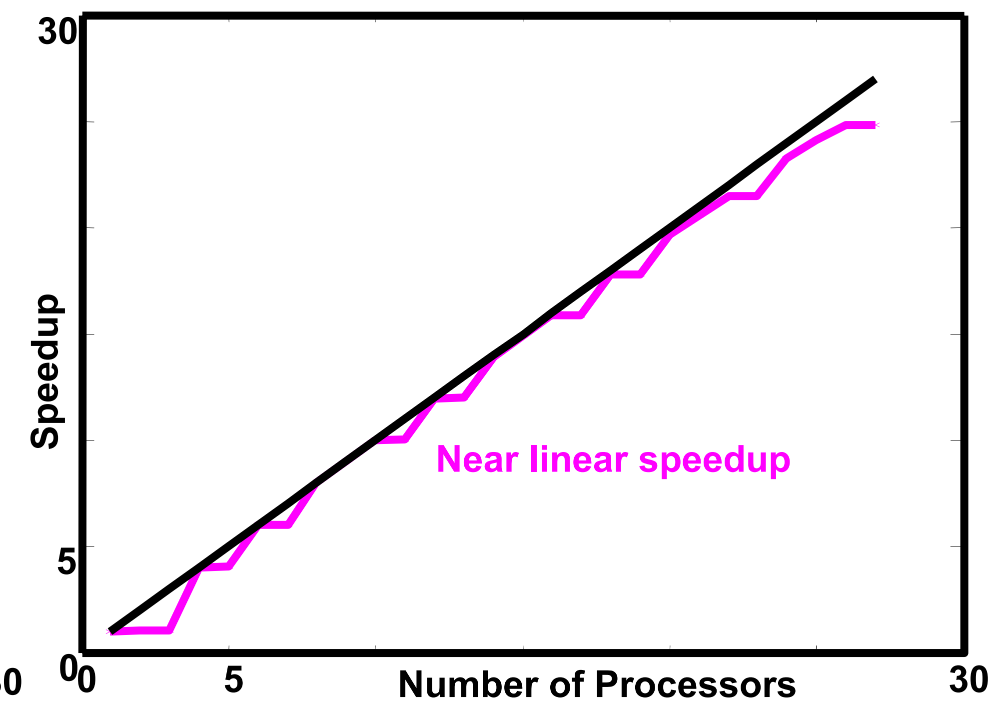
Step 3: Map each node in the stage using the benchmark database (performance model) and the previously best found solution.

Outline

- Motivation and Goals
- Architecture
- **Preliminary Results**
  - **Multi-stage application**
  - **Multi-pipeline application**
- Summary

Multi-stage Application





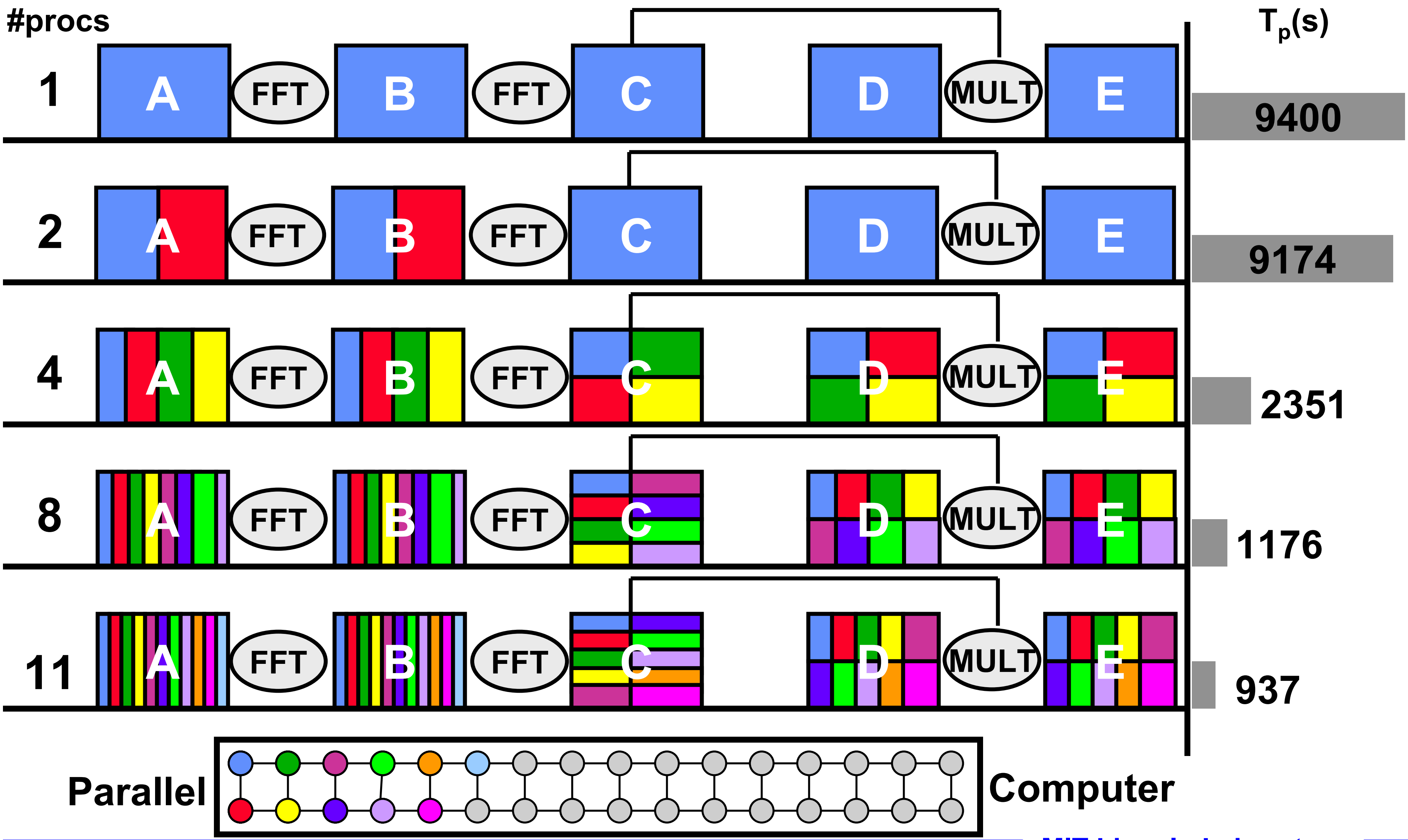

```
A = rand(M,N,p);
B = zeros(M,N,p);
C = zeros(M,N,p);
D = rand(M,N,p);
E = zeros(M,N,p);

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

M,N = 2^{15}

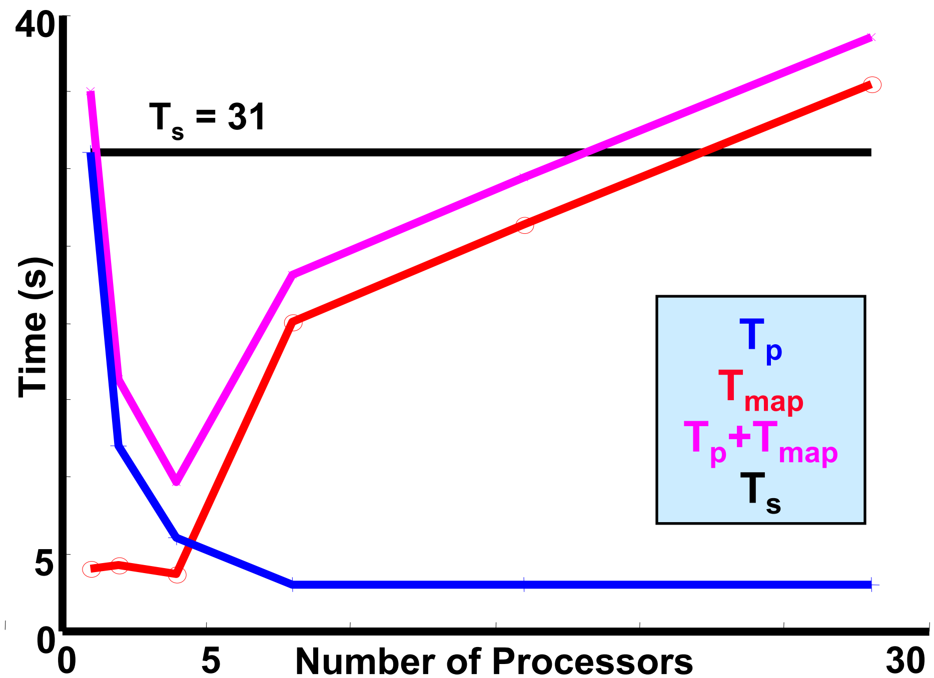
Simulator used for both mapping and execution

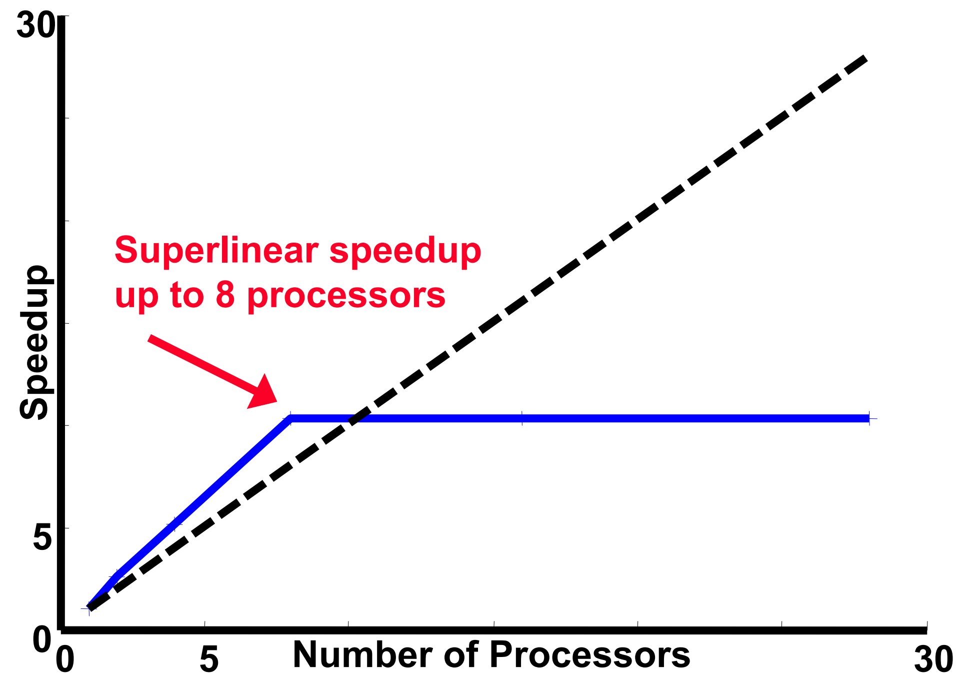
- **Models the underlying architecture**
- **Allows for testing on very large problems**

Multi-stage Application: Output Maps



Multi-pipeline Application





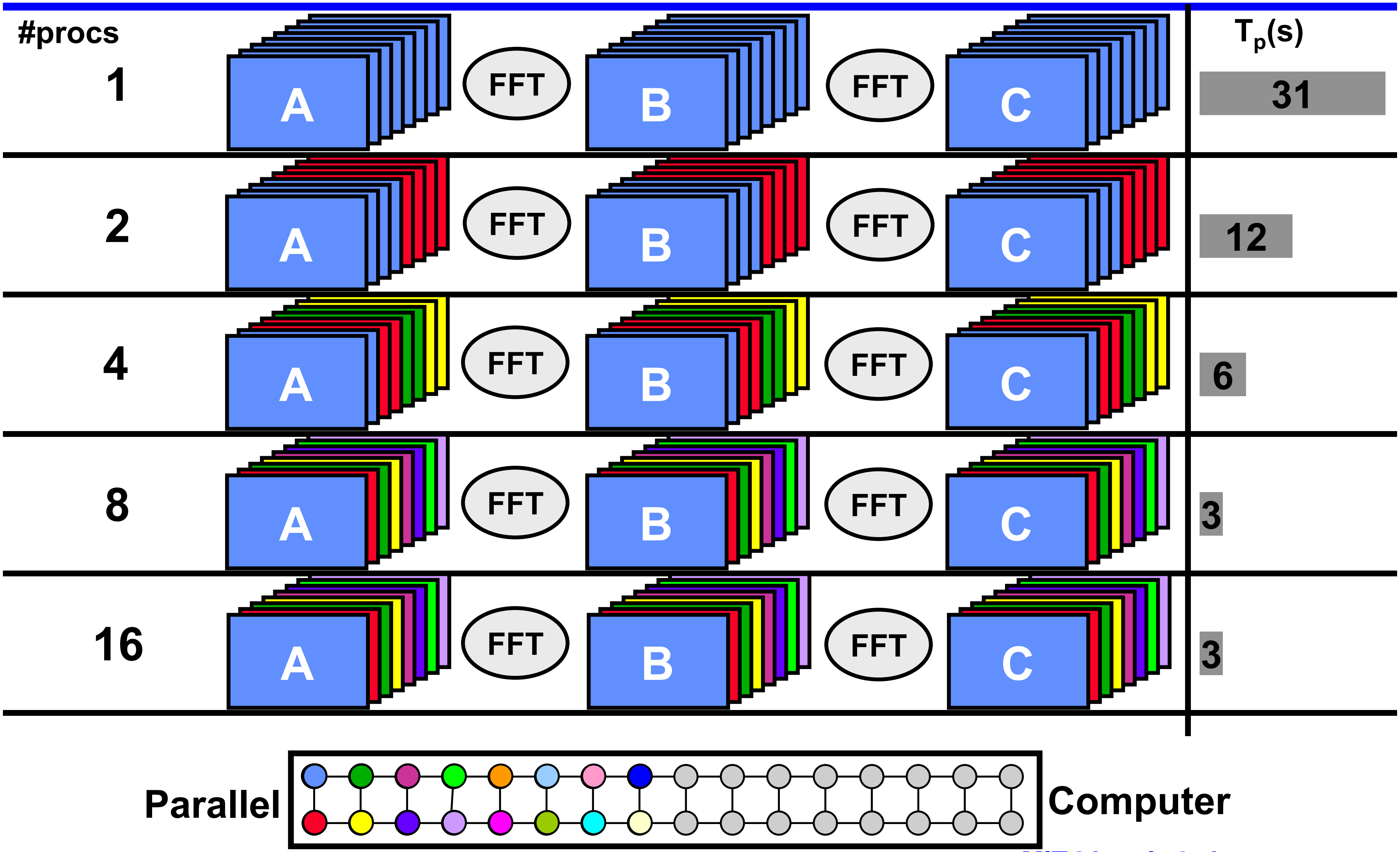

```
A = rand(N,N,M,p);
B = zeros(N,N,M,p);
C = zeros(N,N,M,p);

for i = 1:M
  B(:,:,i) = fft(A(:,:,i),[],1);
  C(:,:,i) = fft(B(:,:,i),[],2);
end
C
```

$N=2^{11}$

$M=2^3$

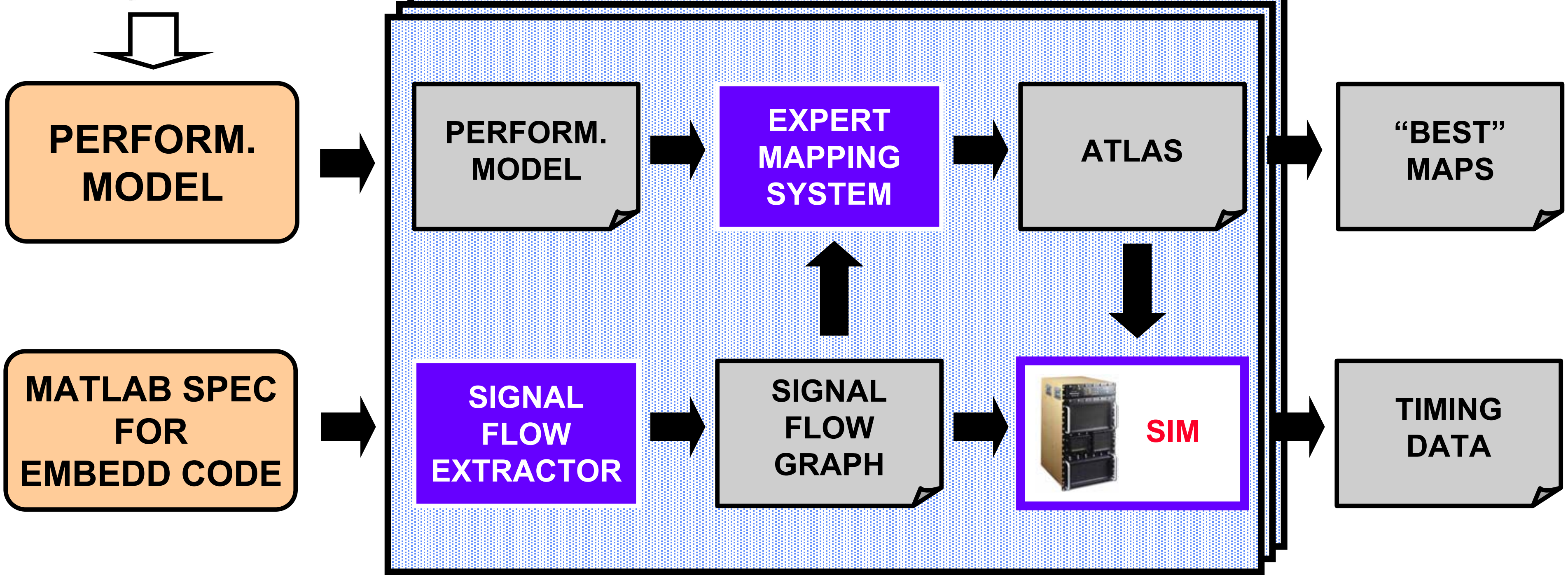
Demonstrated end-to-end functionality

Multi-pipeline Application: Output Maps

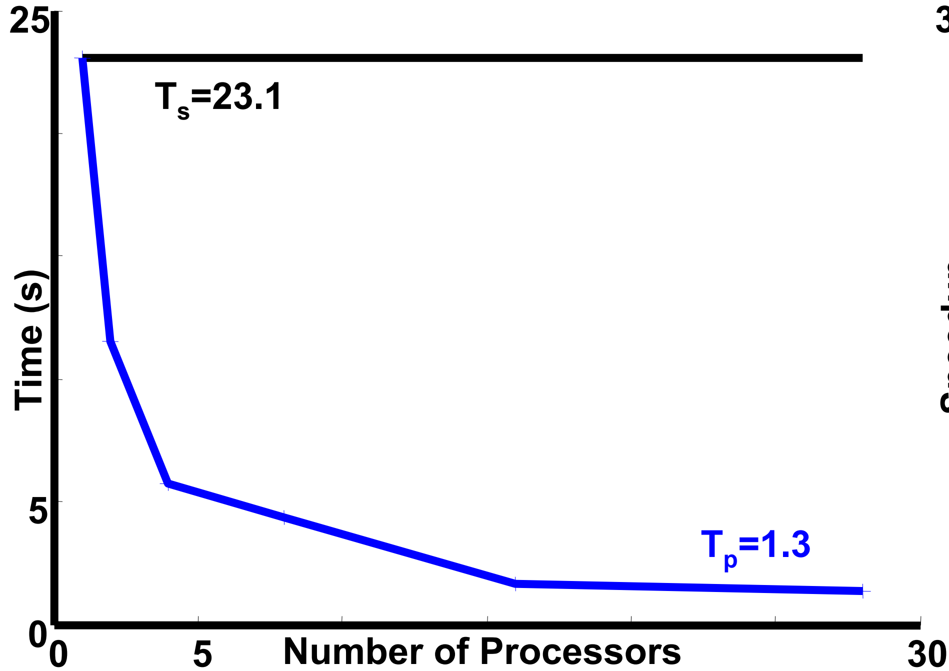


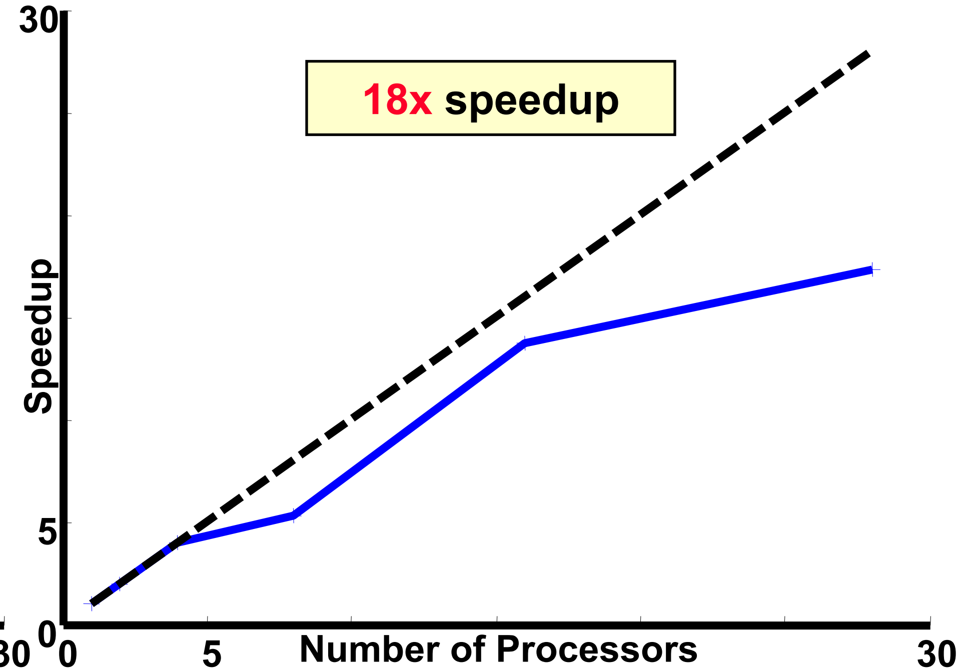
Mapping and Execution
Phase 2

Automatic mapping and performance prediction on an embedded system.



pMapper on Embedded Systems



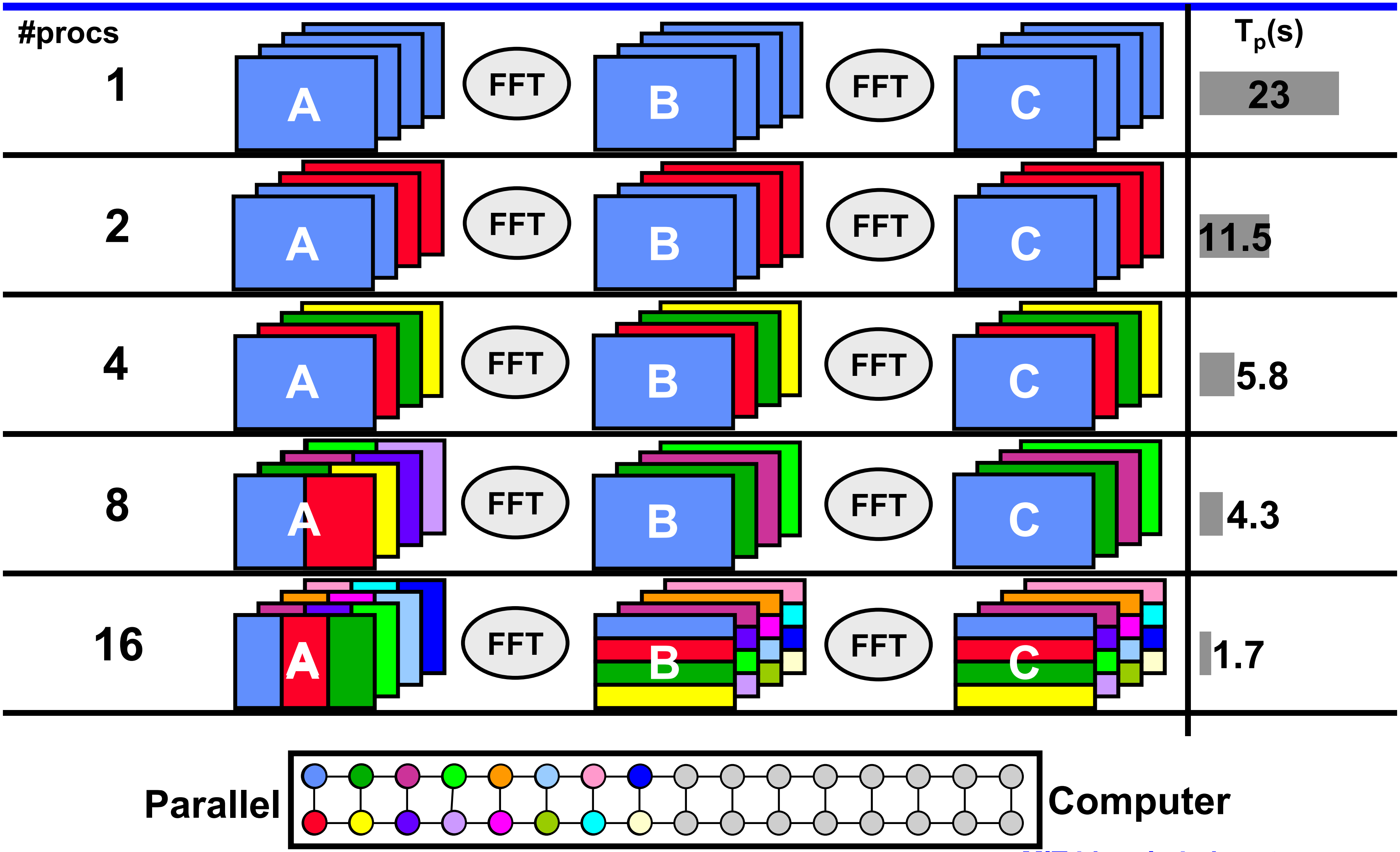



```
A = rand(N,N,M,p);
B = zeros(N,N,M,p);
C = zeros(N,N,M,p);

for i = 1:M
  B(:,:,i) = fft(A(:,:,i),[],1);
  C(:,:,i) = fft(B(:,:,i),[],2);
end
C
```

$N=2^{12}$
$M=2^2$

Explore finer grained architectures
- **Use only simulated timing data for mapping**
- **Execution using simulator**

pMapper on Embedded Systems



Outline

- Motivation and Goals
- Architecture
- Preliminary Results
- **Summary**

Summary

- pMatlab is a parallel Matlab library which is being used widely at Lincoln Laboratory (www.ll.mit.edu/pmatlab)
- pMapper raises the level of abstraction in parallel programming by automatically mapping the analyst's code
- Preliminary experiments have demonstrated both the effectiveness and feasibility of the approach
- pMapper shows promise for mapping applications to embedded systems and future processor architectures

References

[1] A. Petitet, R.C. Whaley, J.J. Dongarra, "Automated Empirical Optimizations of Software and the ATLAS Project," *HPEC 2000 Workshop*.

[2] J. Moura, M. Pueschel, M. Veloso, J.R. Johnson, R.W. Johnson, D. Padua, V. Prasanna, "SPIRAL: Automatic Implementation of Signal Processing Algorithms," *HPEC 2000 Workshop*.

[3] M. Frigo and S.G. Johnson, "FFTW," http://www.fftw.org.

[4] Robert A. van de Geijn, *Using PLAPACK*. The MIT Press, 1997.

[5] Jeremy Kepner and Nadya Travinin, "Parallel Matlab: The next generation," *HPEC 2003 Workshop.*

[6] J. Kepner, "Parallel Programming with MatlabMPI," *HPEC 2001 Workshop.*

[7] J. Kepner, "300x Matlab," *HPEC 2002 Workshop*.

[8] Michael Wolfe, *High Performance Compilers for Parallel Computing*. Addison-Wesley, 1995.

[9] Hank Hoffmann, Jeremy Kepner, Bob Bond, "S3P: Automatic, Optimized Mapping of Signal Processing Applications to Parallel Architectures," *HPEC 2001 Workshop*.

[10] Ron Choy, "Matlab*P", http://supertech.lcs.mit.edu/~cly/matlabp.html.