

---

# Sparse Approximate Inverses in Matlab \*P

Stephanie Taylor

Gurpreetsingh Sachdev

Department of Computer Science

June 2, 2004

# Overview

---

- Introduction to SPAI
- Embarrassingly Parallel Implementation
- Scalable Implementation
- Results and Future Work

# Sparse Approximate Inverse (SPAI)

---

- Consider a sparse matrix  $A$
- We want to find a matrix  $M$  such that
  - $M$  approximates  $A^{-1}$ , and
  - $M$  is as sparse as  $A$ .
- The SPAI algorithm computes  $M$  one column at a time:
  - with a fixed sparsity pattern (easier to implement), or
  - with an adaptive sparsity pattern (more difficult to implement).

# Embarrassingly Parallel Implementation

---

Compute  $M$  one column at a time:

- Computation of each column of  $M$  is entirely independent.
- However, each column may need all of  $A$ .
- The current implementation assigns the computation of certain columns of  $M$  to certain processors, but each processor gets an entire copy of  $A$ .

## Major Advantage

- No communication among processors

Fast

## Major Disadvantage

- Limited by size of  $A$

Not scalable

# Scalable Implementation

---

Handling a distributed  $A$ :

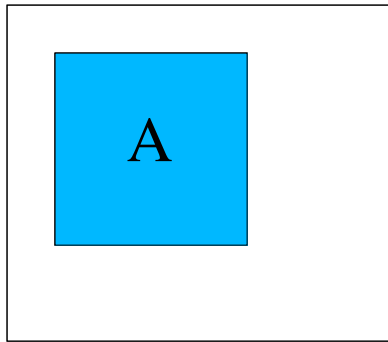
- How do we distribute  $A$  at the beginning?
  - Content-based distribution:
    - This is difficult to do, because we have an adaptive algorithm.
  - Geometric distribution:
    - Simplest option.
- How do we get the correct columns of  $A$  to the appropriate processors during computation?
  - Interprocess communication in \*P mm-mode.
    - There is a Java-based MPI-model implementation in \*P. (Max Goldman and Da Guo at MIT)

# Comparison

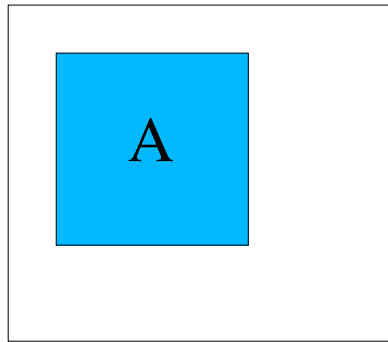
---

## E.P. Implementation

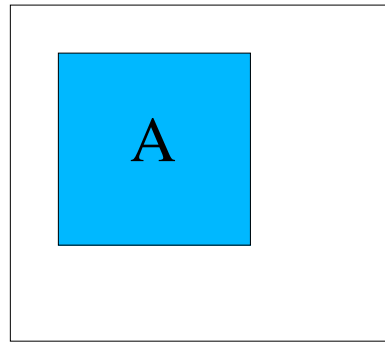
P1



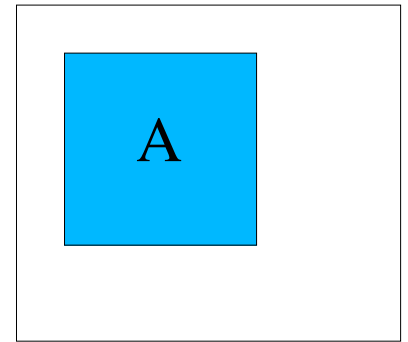
P2



P3

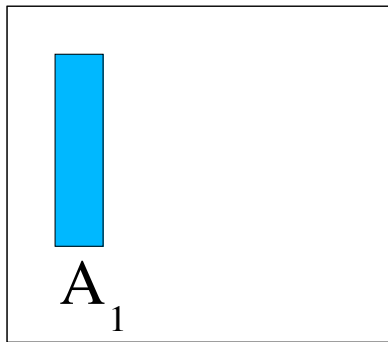


P4

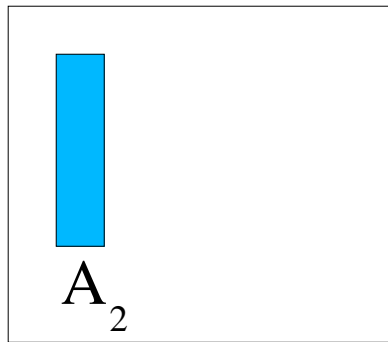


## Scalable Implementation

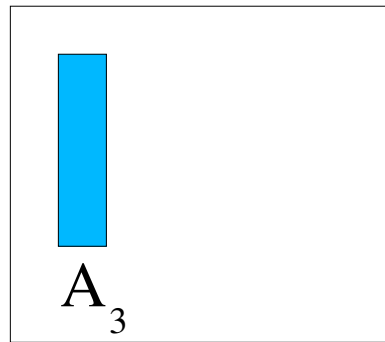
P1



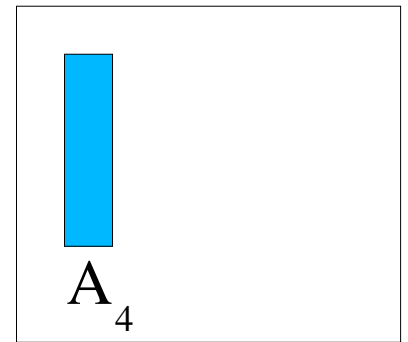
P2



P3

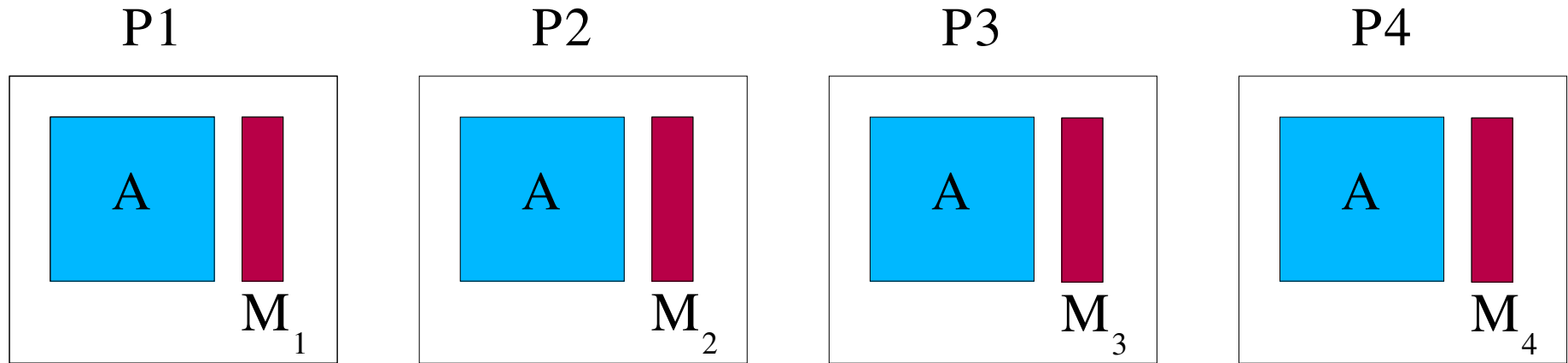


P4

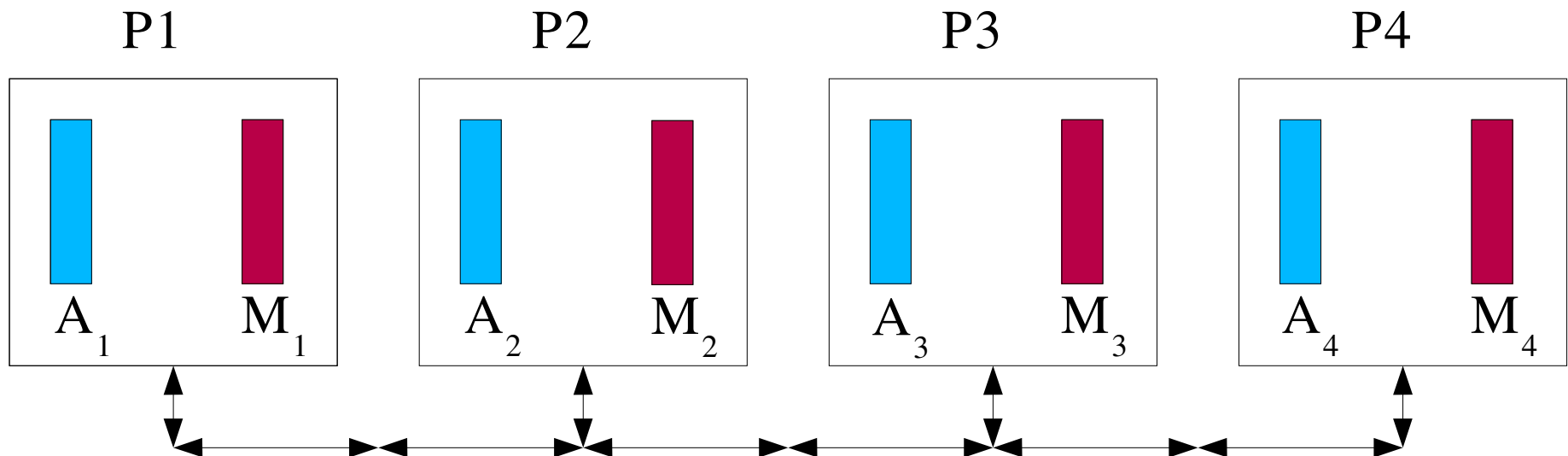


# Comparison

## E.P. Implementation



## Scalable Implementation



# Scalable Implementation Issues

---

- Complicated:

At any time, any processor may need data from any other processor.

- Pi needs to be available to respond to requests
  - if it is performing computation,
  - if it is waiting for responses to its requests, or
  - if it has completed its own computation.

- Slow:

- Introducing communication slows down processing.

- Unstable Infrastructure:

- \*P MPI has never been exercised so vigorously. We had to fix a few bugs.
- \*P MPI does not get along with mm.

# Results and Future Work

---

- Results

- We were able to get it working when the non-zero pattern of  $M$  is fixed:
  - On a smattering of small, random matrices.
  - On an  $886 \times 886$  sparse matrix from the Harwell-Boeing collection: orsirr\_2.
- But it is slow

- Future Work

- Implement Caching
- Handle adaptive sparsity pattern