Design, Implementation and Evaluation of Parallel Pipelined STAP on Parallel Computers

Alok Choudhary

Wei-keng Liao, Donald Weiner, and Pramod Varshney

Richard Linderman and Mark Linderman

ECE Department
Northwestern University

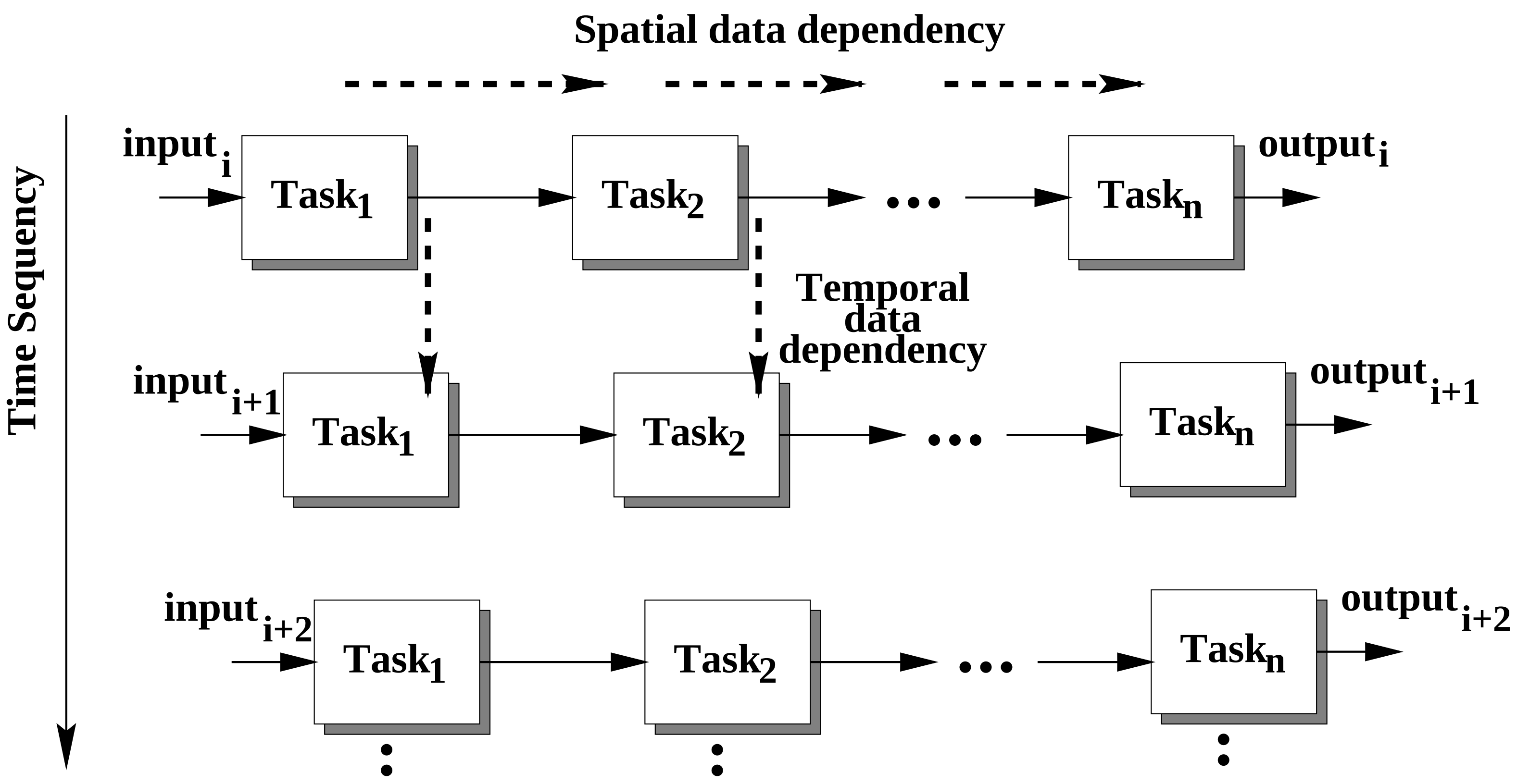
EECS Department
Syracuse University

Air Force Research Laboratory
Information Directorate

Outline

- Parallel Pipeline Computation Model and Characteristics of STAP Algorithms
  - Data Dependencies
  - Throughput and Latency
  - Data Redistribution
  - Processor Assignment
- Design and Implementation
- Performance Results
  - Throughput
  - Latency

Parallel Pipeline Computation Model



Throughput and Latency

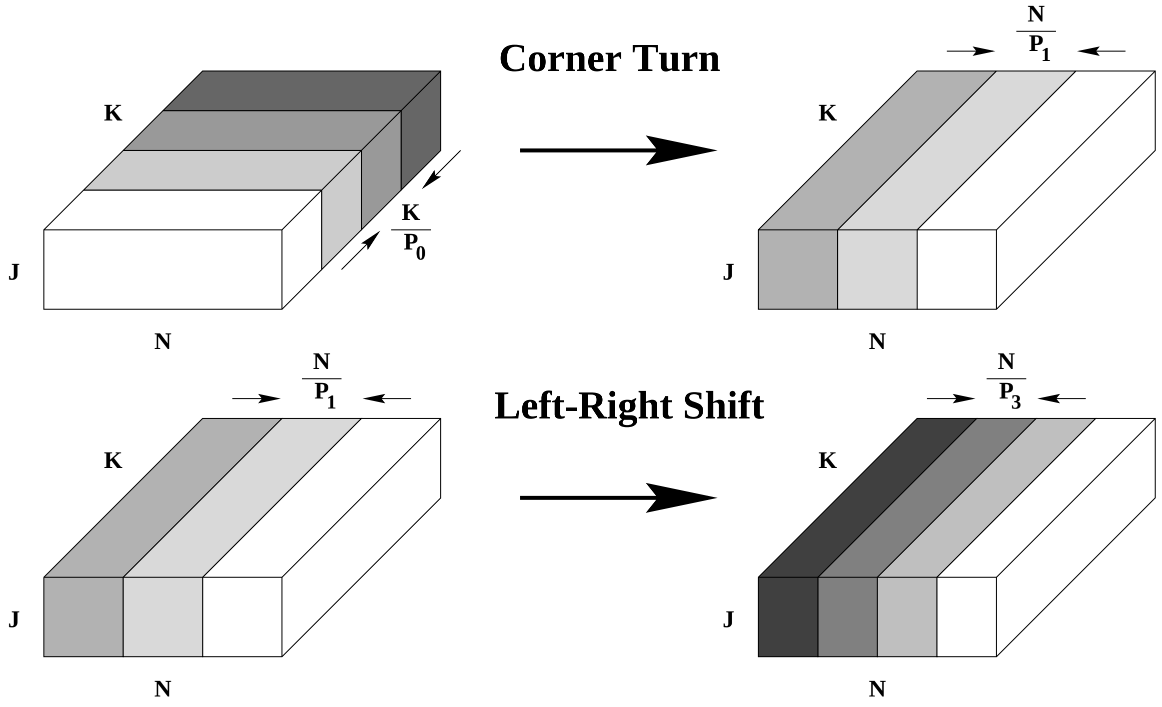
- **Throughput depends on the task with maximal execution time**

$$throughput = \frac{1}{\max\limits_{0 \le i \le n} T_i}$$

- **Latency does not include the temporal data dependency (TTD)**

$$latency = \sum_{i=0}^{n} T_i - \sum_{i \in TTD} T_i$$

Data Redistribution

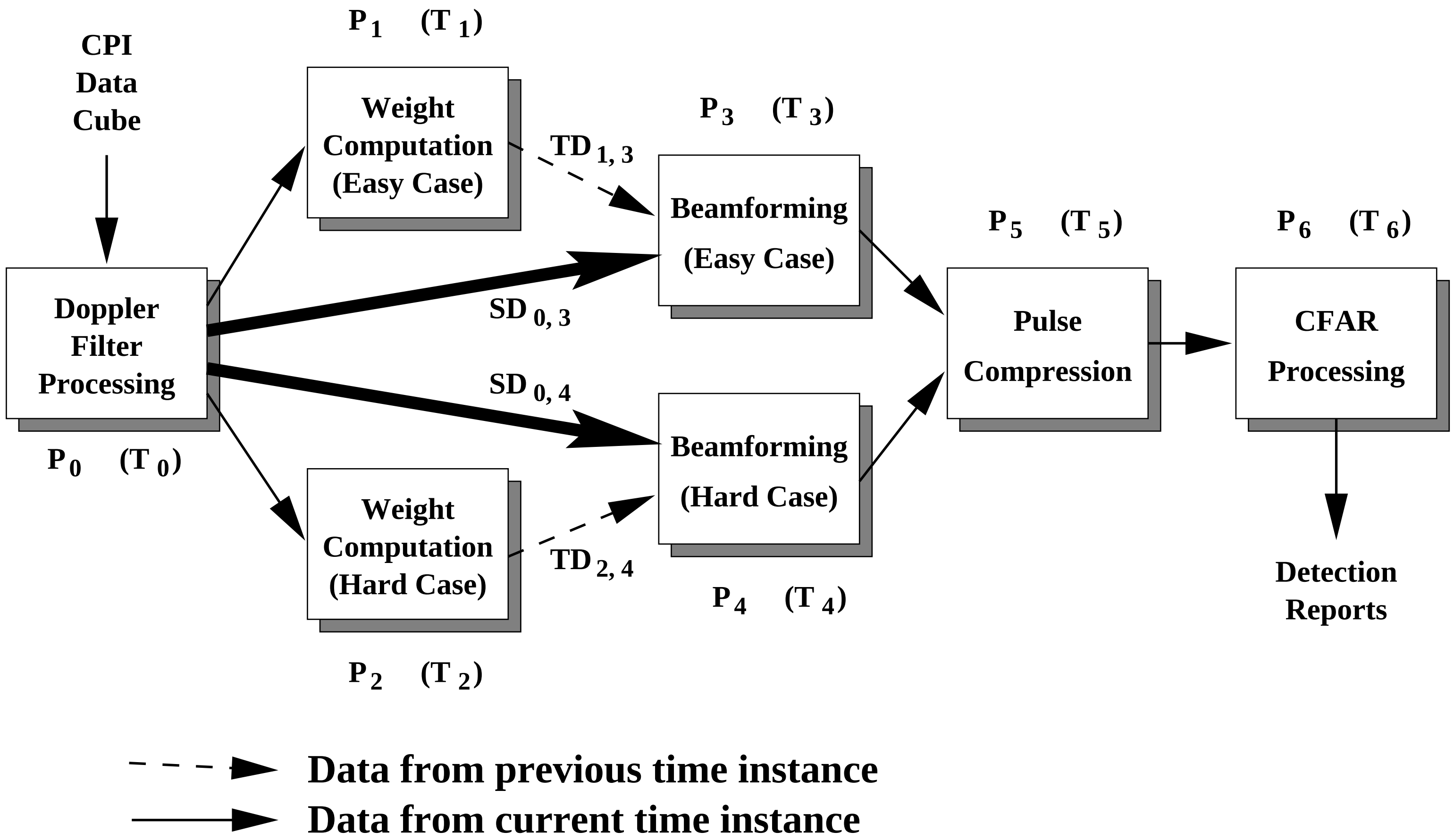


- Intra-task – within the same set of processors
- Inter-task – between two different sets of processors

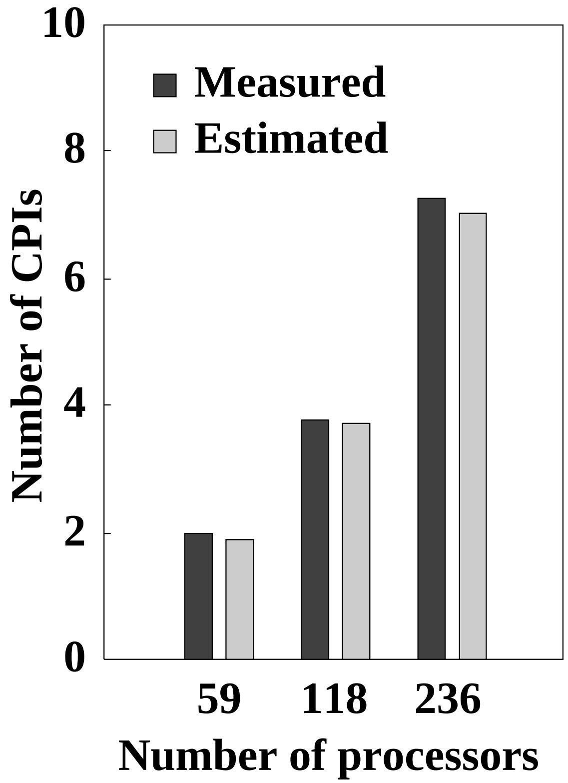
System Platform

- Intel Paragon at the Air Force Research Laboratory (AFRL)
  - 272 compute nodes in a 16×17 rectangle mesh
  - processor is an i860 Intel RISC CPU with 64 MB RAM
  - peak performance for one processor is 100 MFLOP

Implementation and Data Flow

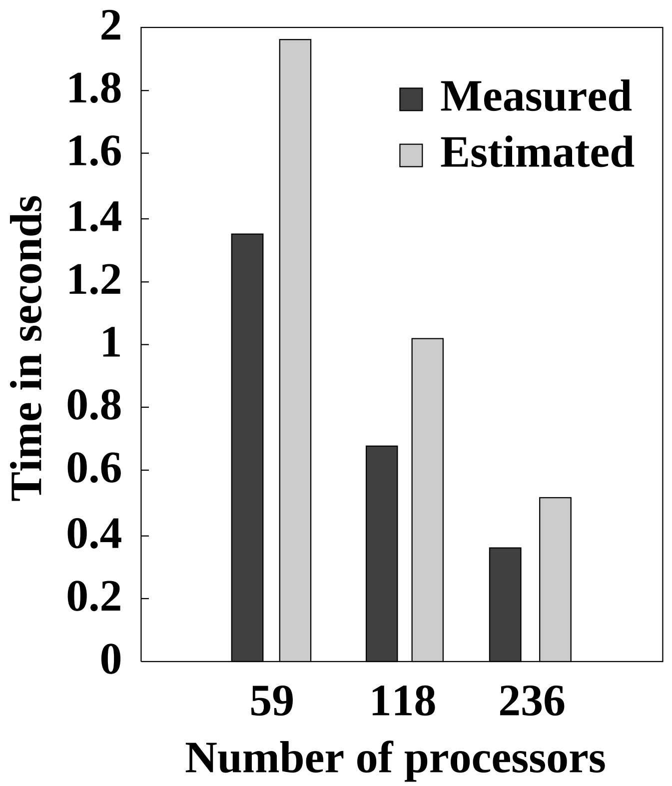


Summary of Throughput



- Good scalability and speedups
- Run on a large configuration
- Measured and estimated values very close
- Throughput of **7.2** CPIs per second
- Base case processor assignment
  - $P_0 = 8, P_1 = 4, P_2 = 28, P_3 = 4$
  - $P_4 = 7, P_5 = 4, P_6 = 4$
  - NOP are $2\times$ and $4\times$ in other 2 cases

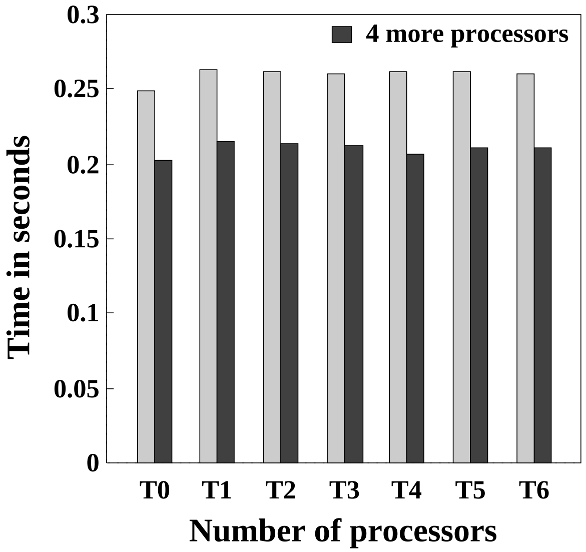
Summary of Latency



- Numbers corresponding to the previous throughput chart
- Good scalability and speedups
- Measured latency is smaller because overlap between computation and communication achieved
- Latency of 360 ms per CPI

Adding 4 More Processors to Case 2

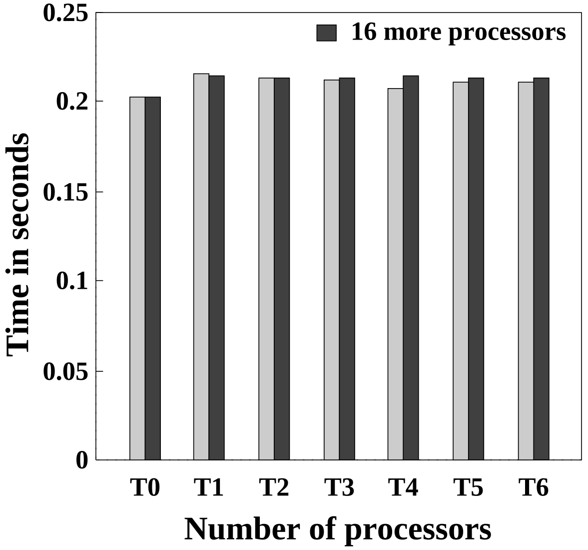
# of proc	118	122
throughput	3.80	5.02
latency	0.68	0.55



- Add 4 more processor Doppler filter task
- Total No. of processors increased by 3%
- Throughput increased by 32%
- Latency reduced by 19%
- Communication amount between Doppler filter task and its successor tasks is reduced
- Adding more processors to one task affects performance of other tasks
- Such effects are very difficult to capture in purely theoretical models because of the secondary effects

Adding 16 More Processors

# of proc	122	138
throughput	5.02	4.91
latency	0.55	0.42



- 16 extra processors available and add to Pulse compression and CFAR tasks
- Total No. of processors is increased by 14%
- Throughput is not increased much
- Weight tasks are bottleneck tasks, all successor tasks wait for its completion
- Latency reduced by 23%
- Computation time is reduced in last two tasks
- Tradeoff occurs when adding more processors to increase throughput or to reduce latency

Conclusion

- Parallel pipeline computation model on commercial HPC systems
- Modular, scalable, and portable software design
- Real full application parallelized and implemented from AFRL
- Scalable performance on both throughput and latency
- Further optimizations
  - Inter-task data redistribution
  - Processor assignments