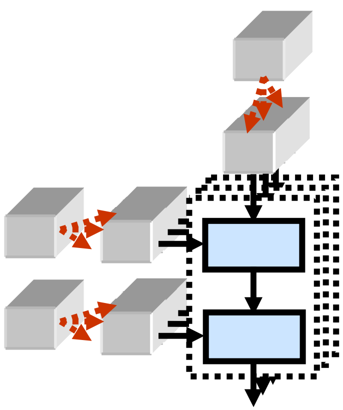
Parallel processing is not easy

Good schedules for parallel implementations of composite tasks are hard to find.

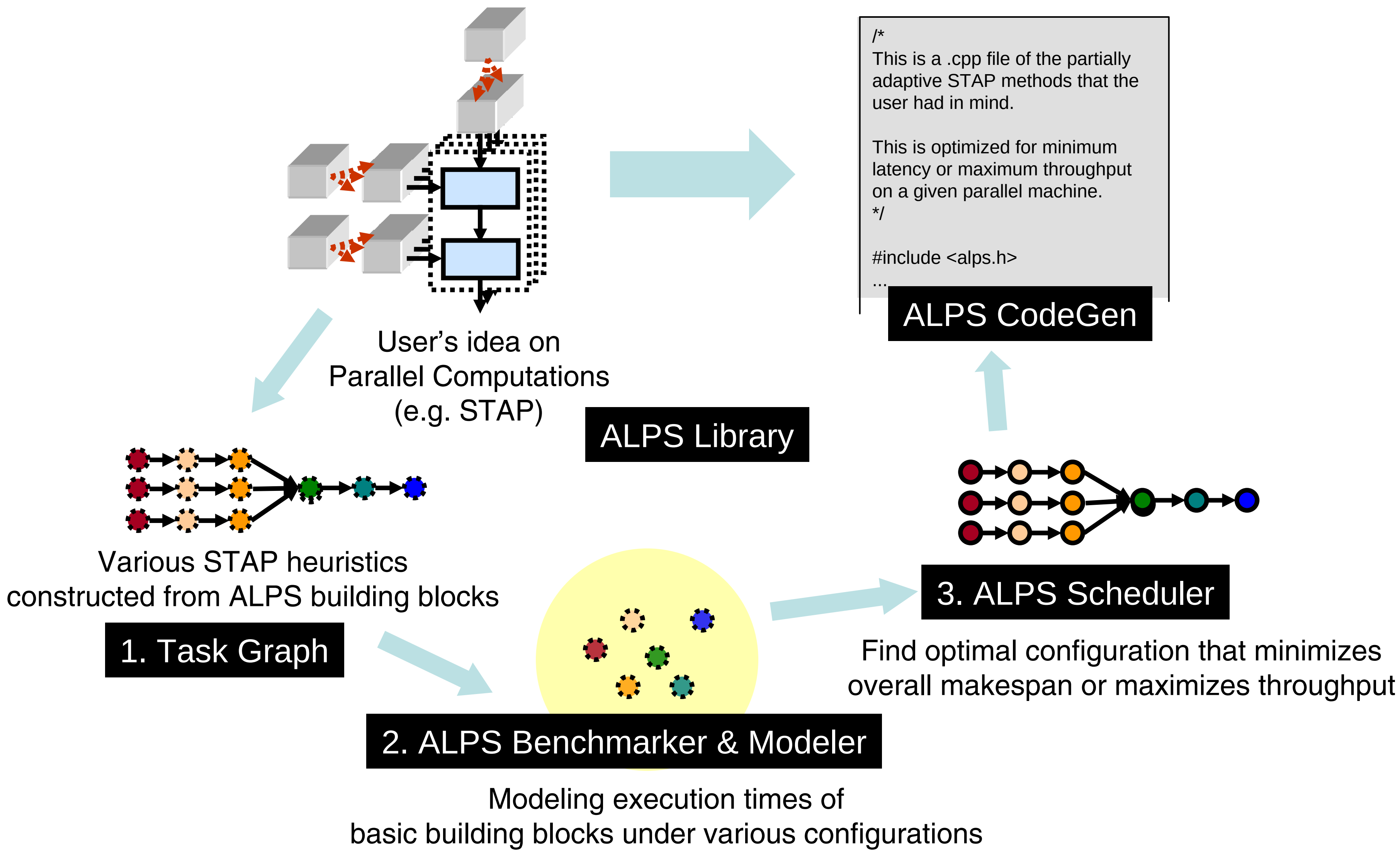


User's idea on
Parallel Computations
(e.g. STAP)

- Parallel implementation
- Mapping and scheduling
- Optimization for makespan or throughput

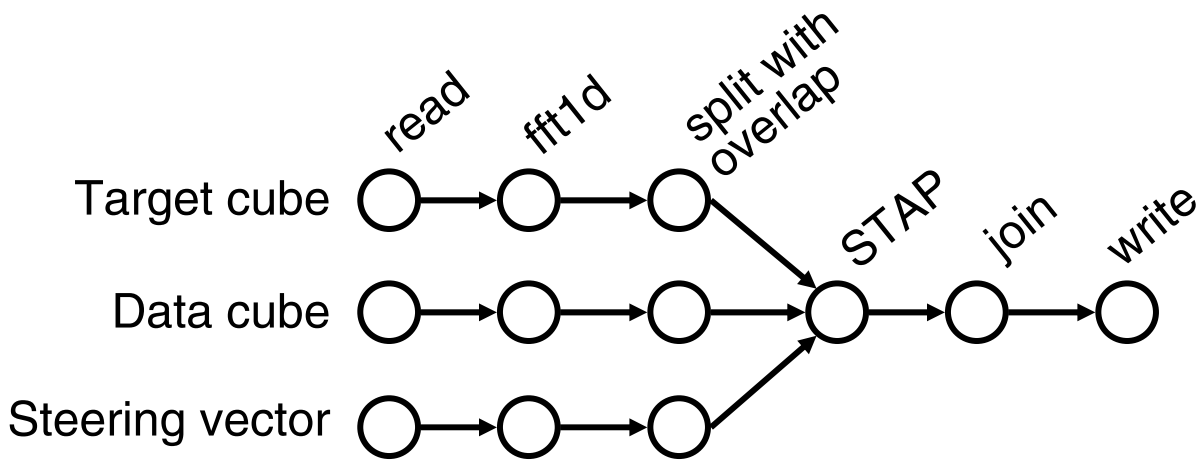
Executable file
for a given parallel computer

Parallel processing can be made easy with ALPS software framework



1 Task Graph: PRI-overlapped STAP

User describes the **algorithm** using task graphs



ALPS software framework

determines the **implementation parameters** such as number of processors for each task and return the **executable code**

2 ALPS Benchmarker and Modeler

1. Measures of actual timings y_i for a set of **input parameters**
2. **Candidate terms** $\{g_k\}$ such as the size of data cube or the size of processor cube are generated based on the input parameters
3. Model coefficients x_k's are found using least-squares

$$T_{task} \approx \sum (x_k g_k)$$

$$\min_x \left\| W(Ax-b) \right\|_2^2 = \min_x \left\| W \begin{pmatrix} g_1 & g_2 & g_3 & \cdots & g_{n-1} & g_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} - W \begin{pmatrix} \overline{y}_1 \\ \overline{y}_2 \\ \vdots \\ \overline{y}_m \end{pmatrix} \right\|_2^2$$

Solved incrementally by adding a best column to the model at a time.

$$\min_x \| W(Ax-b) \|_2^2$$

Ordinary least-squares

W = I

COV-weighted least-squares (BLUE estimator)

$W = R^{-1/2}$ where R is the covariance matrix

one of the best, but requires lots of replications hence not practical

VAR-weighted least-squares

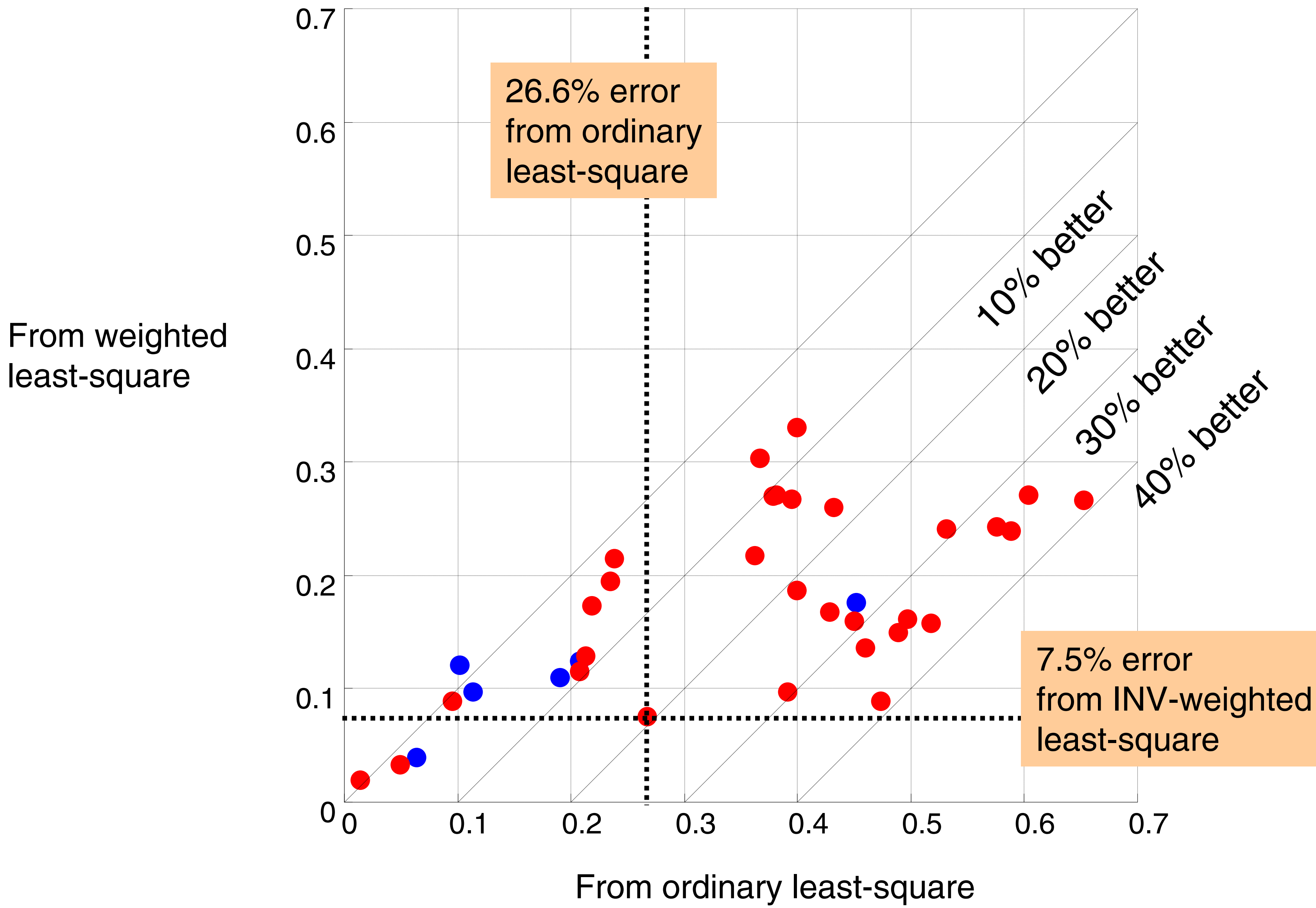
$W = D^{-1/2}$ where D=diag(R)

INV-weighted least-squares

W = diag(1/b)

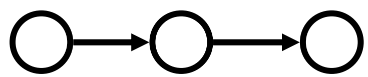
Used ordinary, VAR-weighted and INV-weighted least-squares with 5 replications for each measurement

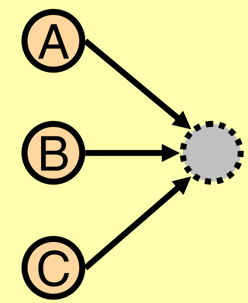
$$(\text{mean of relative error}) = \frac{1}{N}\sum_{i}^{N}\frac{|\mathbf{a}_i\mathbf{x} - \bar{y}_i|}{\bar{y}_i}$$



3 ALPS Scheduler

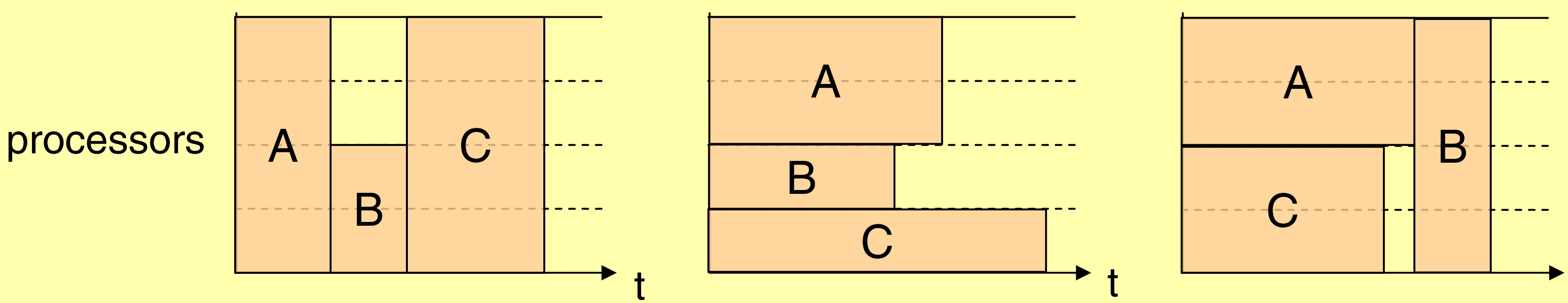
Schedules tree-shaped task graphs





(Tree-shaped graphs) = (linear chains) + (joining nodes)

rather simple — complicated



Data-parallelism only [3]

Task-parallelism preferred [4]

Other possibly better schedules

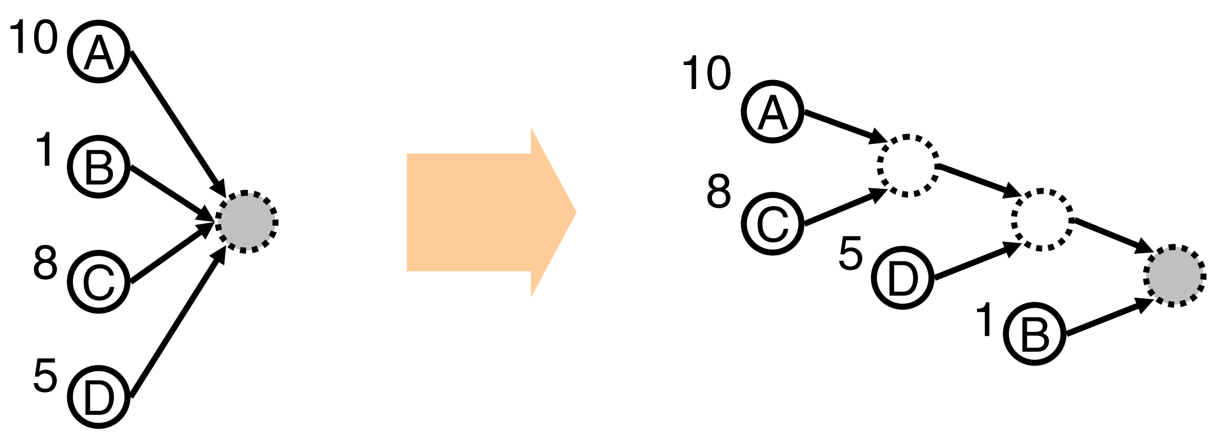
We consider both data-parallel (series) and task-parallel (parallel) schedules for a joining node

Scheduling a joining node with **two** incoming arcs

- Use dynamic programming considering two incoming arcs in series and in parallel

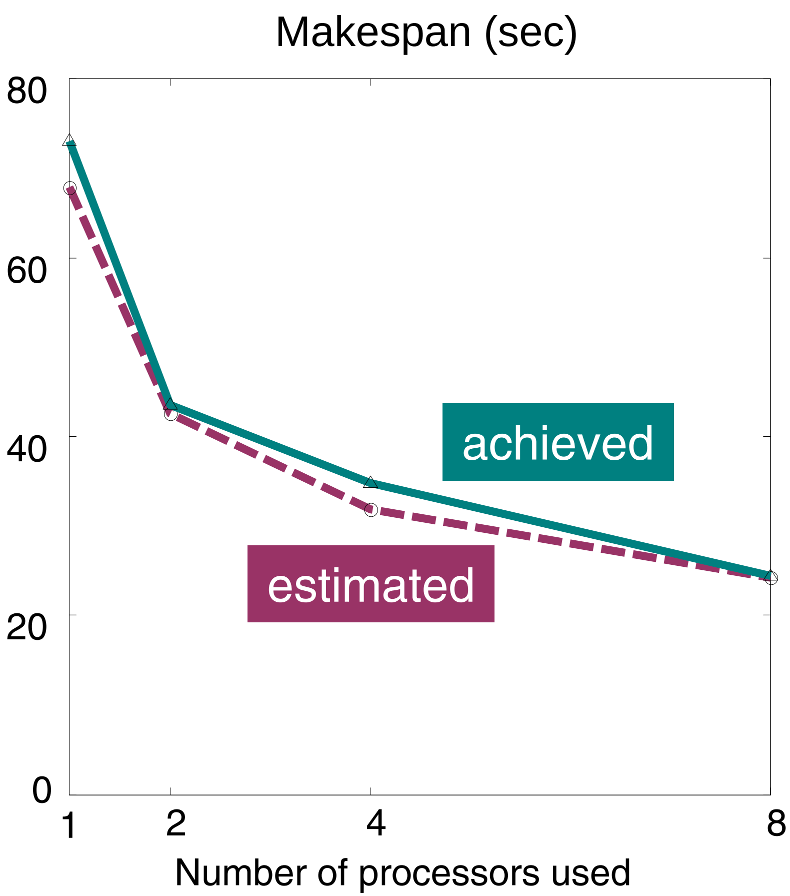
Scheduling a joining node with **more than two** incoming arcs

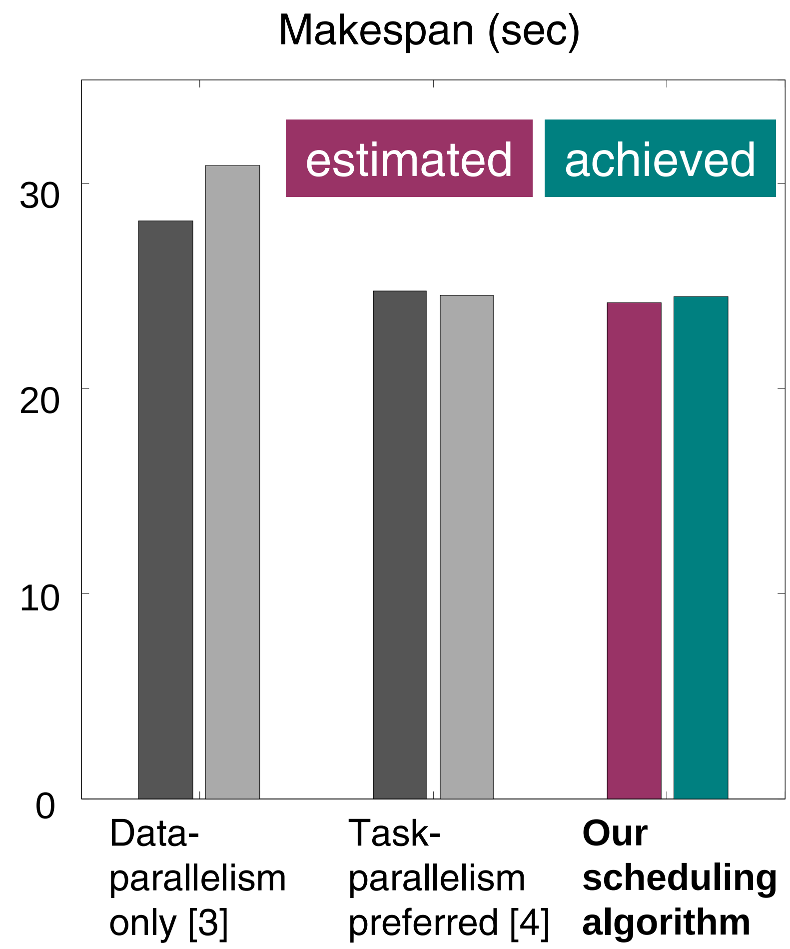
1. Sort the incoming arcs based on makespan when maximally parallelized
2. Schedule two incoming nodes at a time



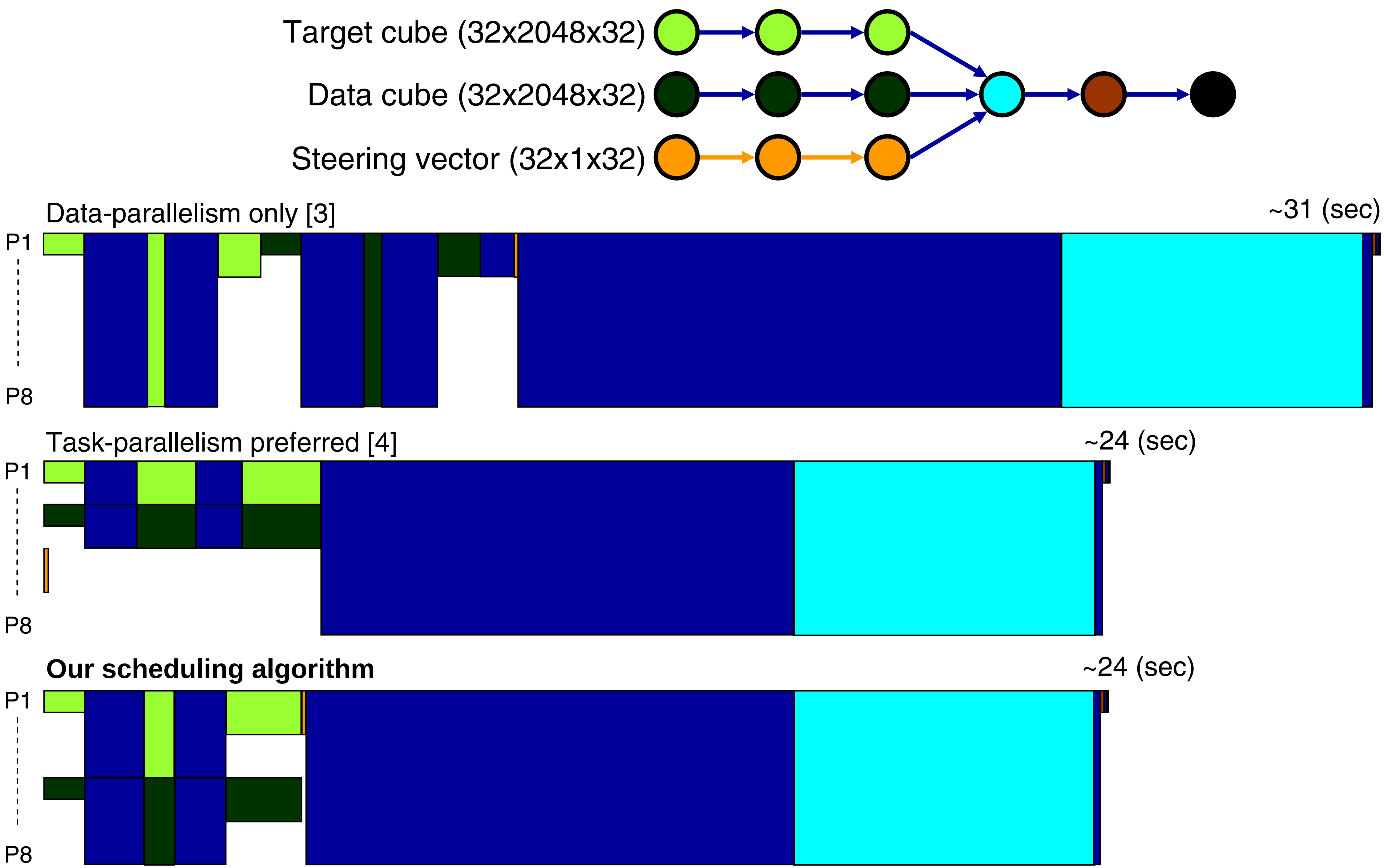
Experiments on 8-node linux cluster:
executable codes were generated by ALPS software framework

Makespan: achieved vs. estimated

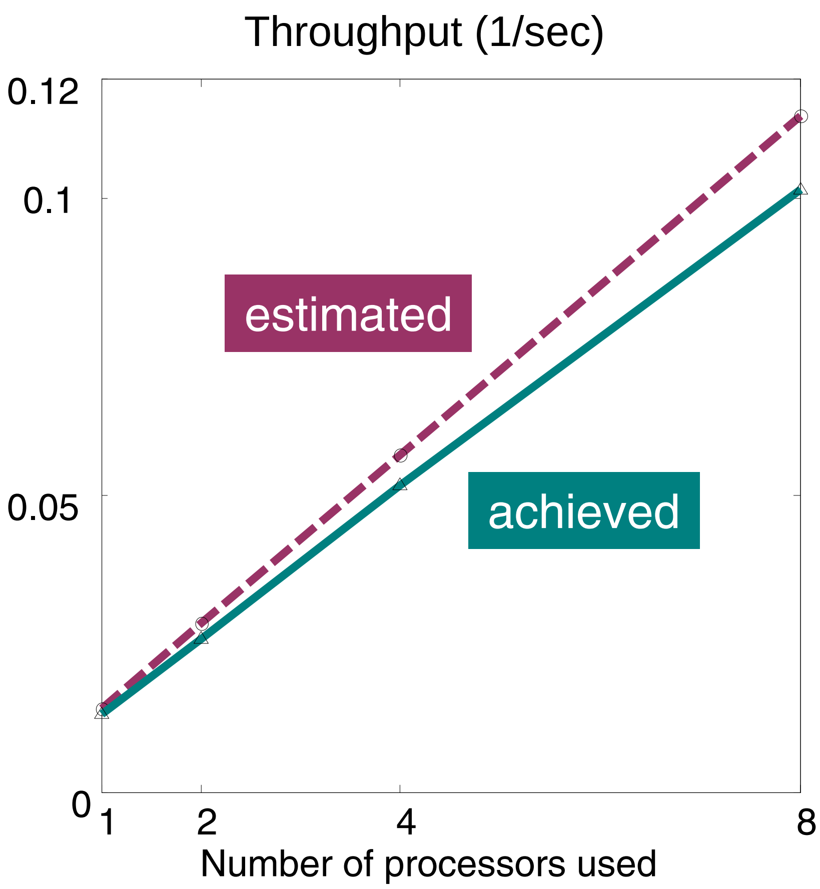


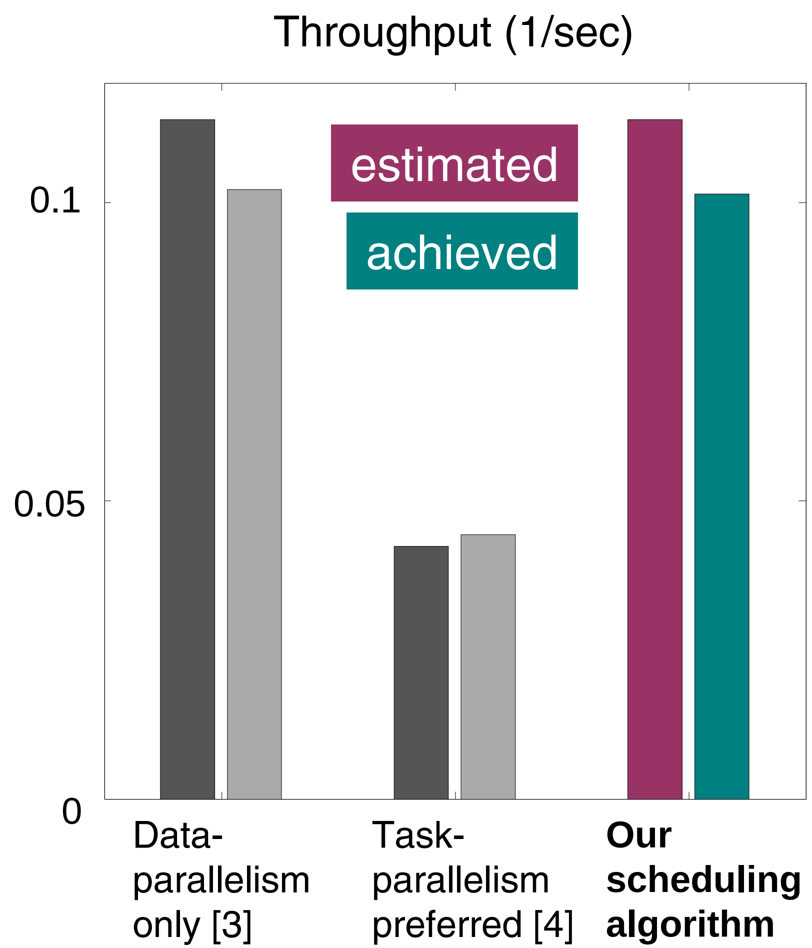


Schedules found in makespan minimization

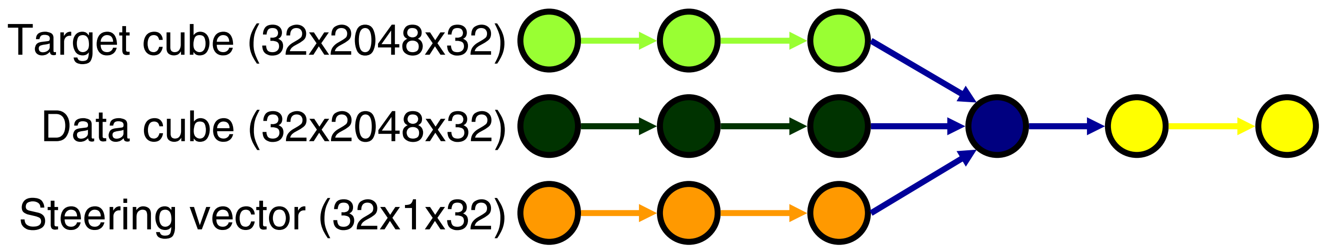


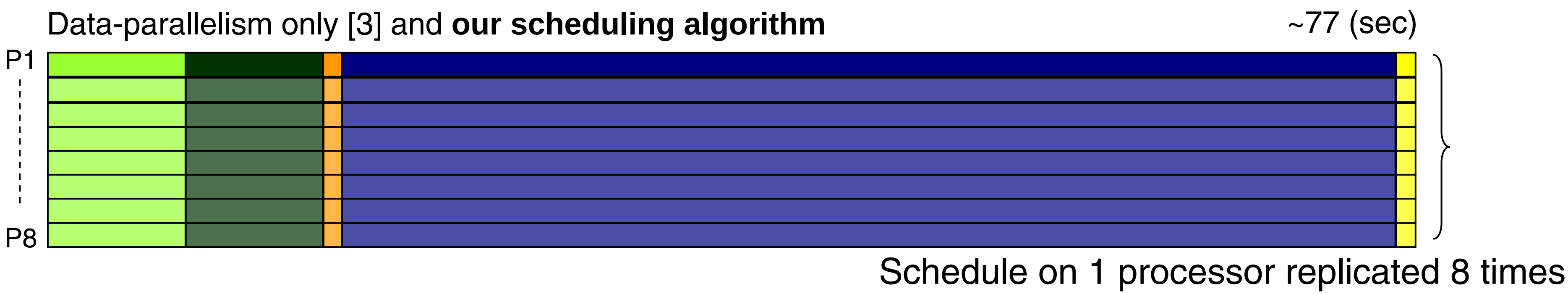
Throughput: achieved vs. estimated

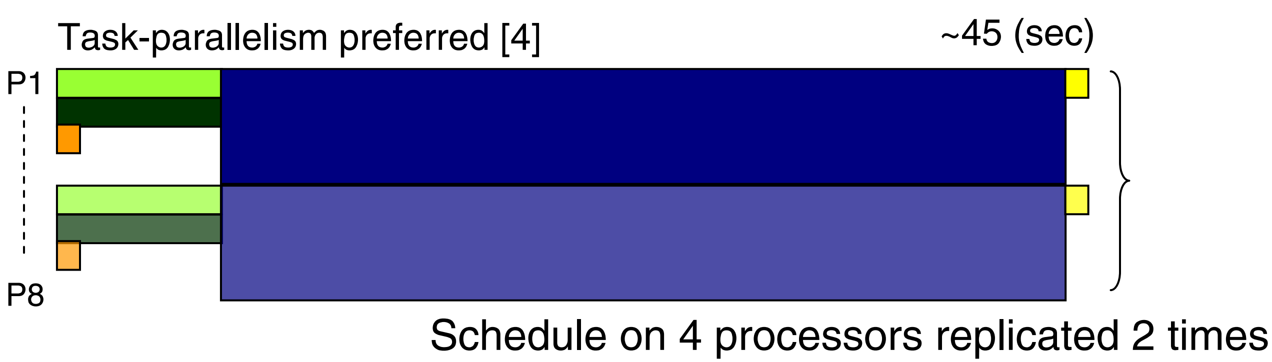




Schedules found in throughput maximization







Conclusion

The schedules found by ALPS framework are as good as or often better than those computed by other published algorithms

The relative error between the estimated and the achieved makespan/throughput were around 10% confirming that the modeled execution time is useful in predicting performance

Weighted least-squares produced estimates 20—30% better than ordinary least-squares in terms of relative errors; may be useful when modeling the unreliable timings.

Related works

Frameworks

[1] **PVL** and [2] **S^{3}P**

> We use text version of task graph as front-end and source code generation as back-end without users dealing with C++ code
>
> We cover larger scheduling problem space (tree-structured task graphs) than S^{3}P.

Scheduling algorithms

[3] J. Subholk and G. Vondran: chain of tasks only

[4] D. Nicol, R. Simha, and A. Choudhary: s-p task graph, but task-parallelism is preferred

(… and many other works; refer to reference in the abstract)

References

[1] Eddie Rutledge and Jeremy Kepner, “**PVL**: An Object Oriented Software Library for Parallel Signal Processing”, *IEEE Cluster* 2001.

[2] H. Hoffmann, J. V. Kepner, and R. A. Bond, “**S3P**: automatic, optimized mapping of signal processing applications to parallel architectures,” in *Proceedings of the Fifth Annual High-Performance Embedded Computing (HPEC) Workshop*, Nov. 2001.

[3] J. Subhlok and G. Vondran, “Optimal Use of Mixed Task and Data Parallelism for Pipelined Computations”, *Journal of Parallel and Distributed Computing*, vol. 60, 2000.

[4] D. Nicol, R. Simha, and A. Choudhary, “Optimal Processor Assignments for a Class of Pipelined Computations”, *IEEE Transaction on Parallel and Distributed Systems*, vol. 5(4), 1994.