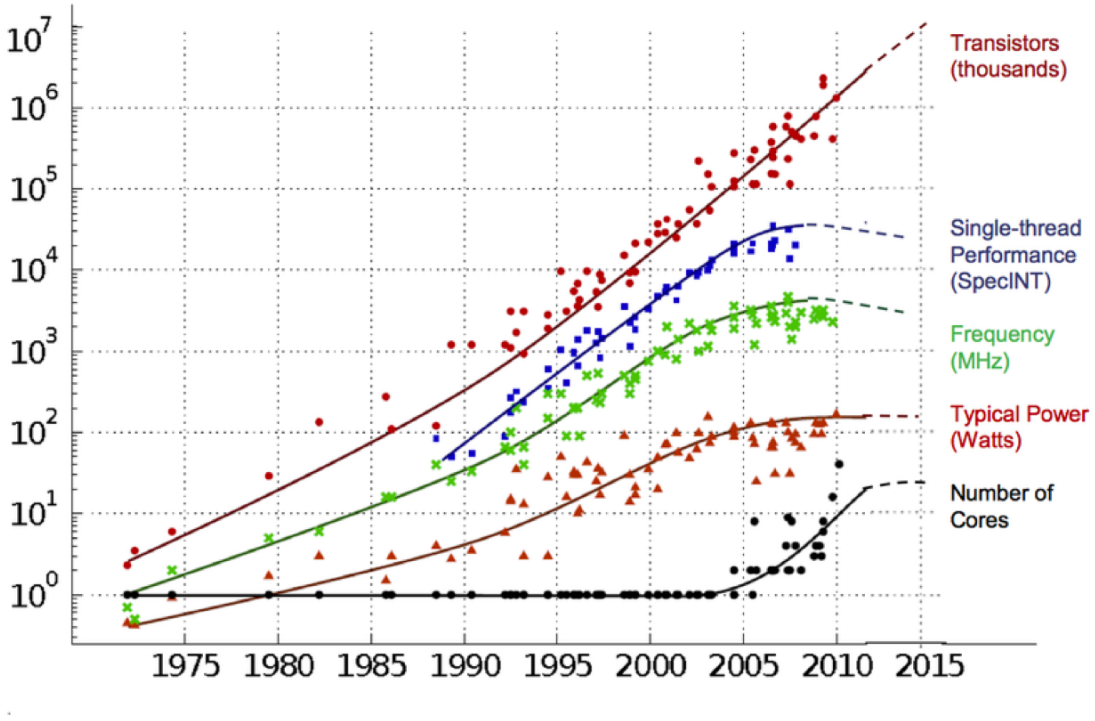
GraphLab: A New Framework For Parallel Machine Learning

YUCHENG LOW, JOSEPH GONZALEZ, AAPO KYROLA, DANNY BICKSON, CARLOS GUESTRIN, JOSEPH HELLERSTEIN

Presented by Hyun Ryong (Ryan) Lee

Parallel Programming is Important for ML

End of frequency scaling -> Need parallelism to scale



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

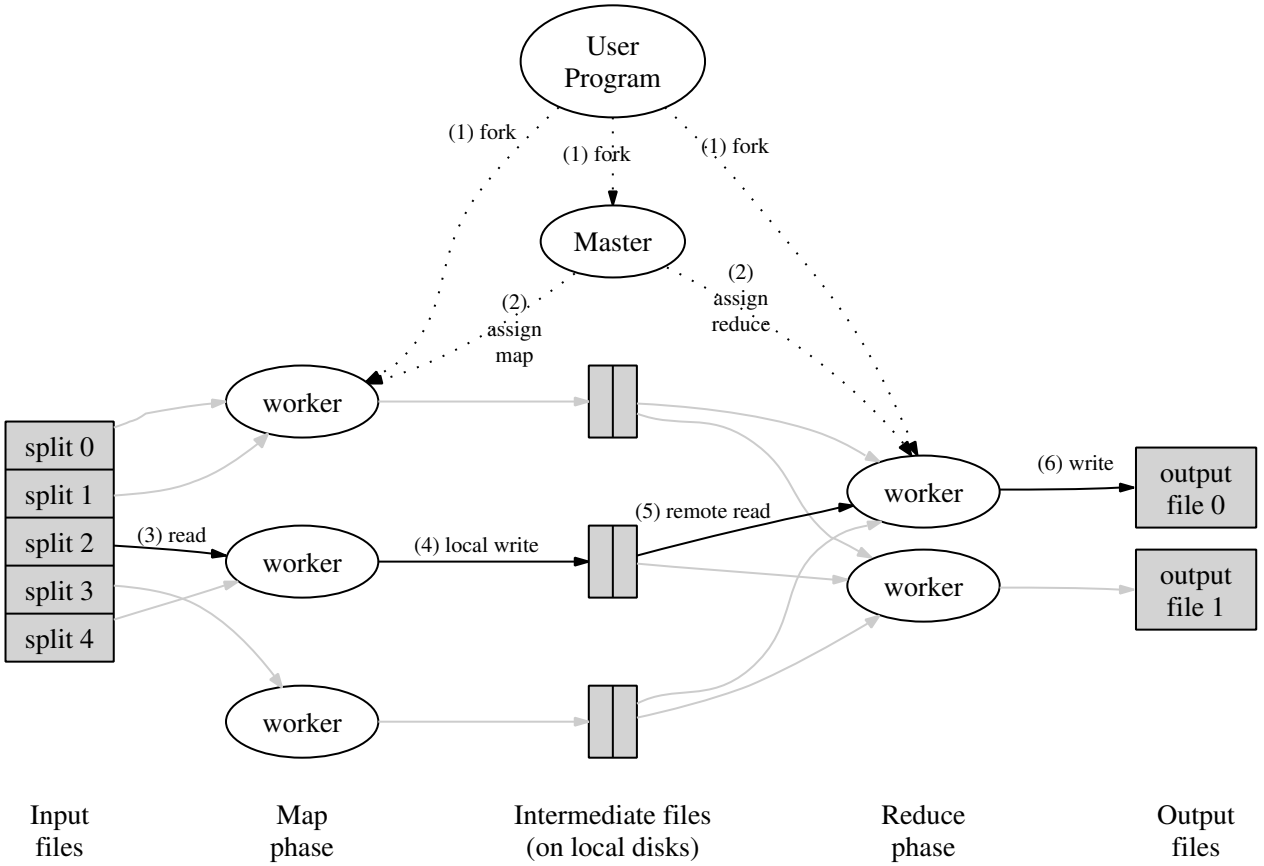
Existing Frameworks are Unsuitable for ML

High-Level Framework: MapReduce

- Limited Scope: Targets "embarrasingly parallel" applications
  - Many ML algorithms have *data dependences (Belief propagation, gradient descent, ...)*

- Cannot express *Iterative algorithms* effectively
  - Many ML algorithms are *iterative*

Existing Frameworks are Unsuitable for ML



Existing Frameworks are Unsuitable for ML

Low-level frameworks: Pthreads, MPI

- DAG abstraction: nodes are computations, edges are data flow
- Expressive, but *hard to program*
  - *Reason about synchronization*
  - *Load balancing*
  - *Deadlock, Livelock, …*

GraphLab

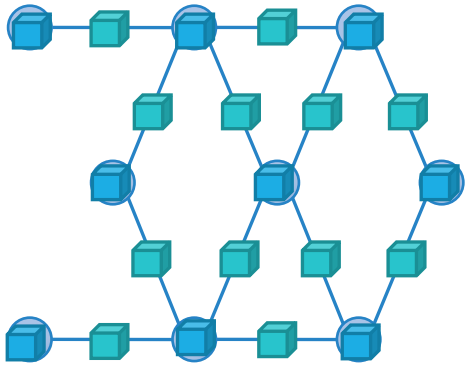
A *vertex-centric, asynchronous, shared memory* abstraction for graph processing

- Updates are vertex-centric
- No barrier synchronization
- No message passing abstraction

GraphLab Overview

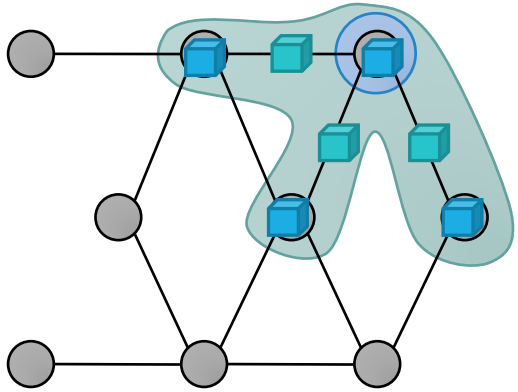
Graph Based

Data Representation

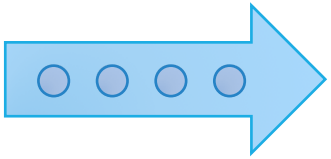


Update Functions

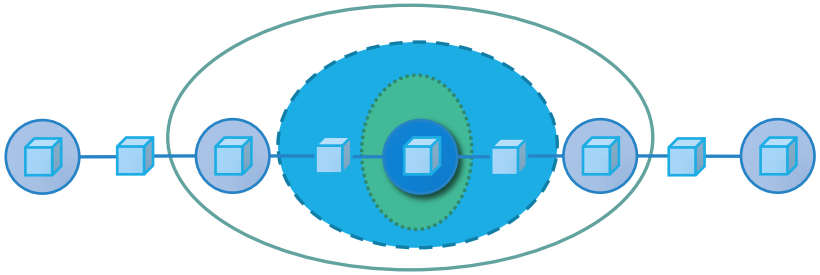
User Computation



Scheduler

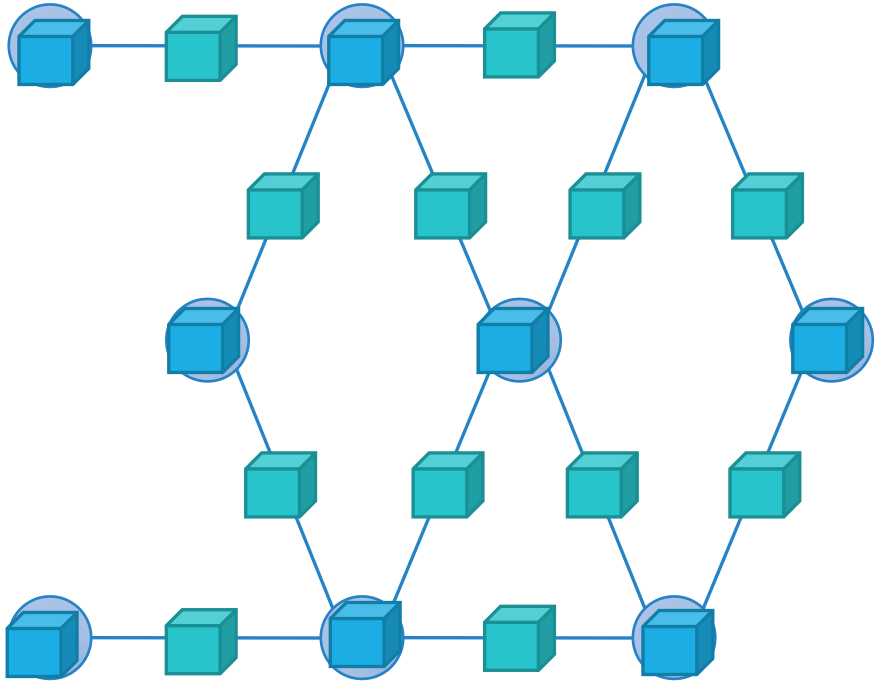


Consistency Model



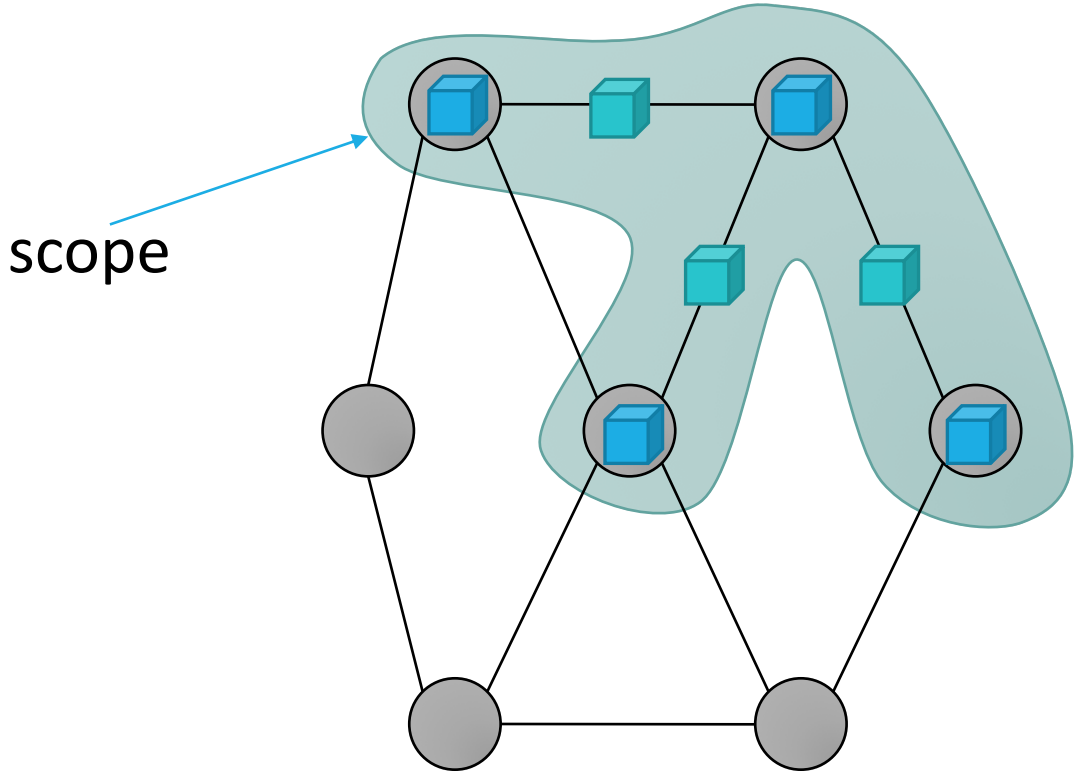
Data Model

Data modeled as a *graph* with arbitrary data associated with each vertex and edge



Local Updates

Update function: defines local computation on a *scope* of a vertex



Sync Mechanisms

Much like fold/reduce

- Fold: aggregate data sequentially
- Merge: parallel tree reduction of folded data
- Apply: apply updated data to global *shared data table(SDT)*

Either *periodic*, or triggered by an update

Data Consistency

Need to resolve race conditions

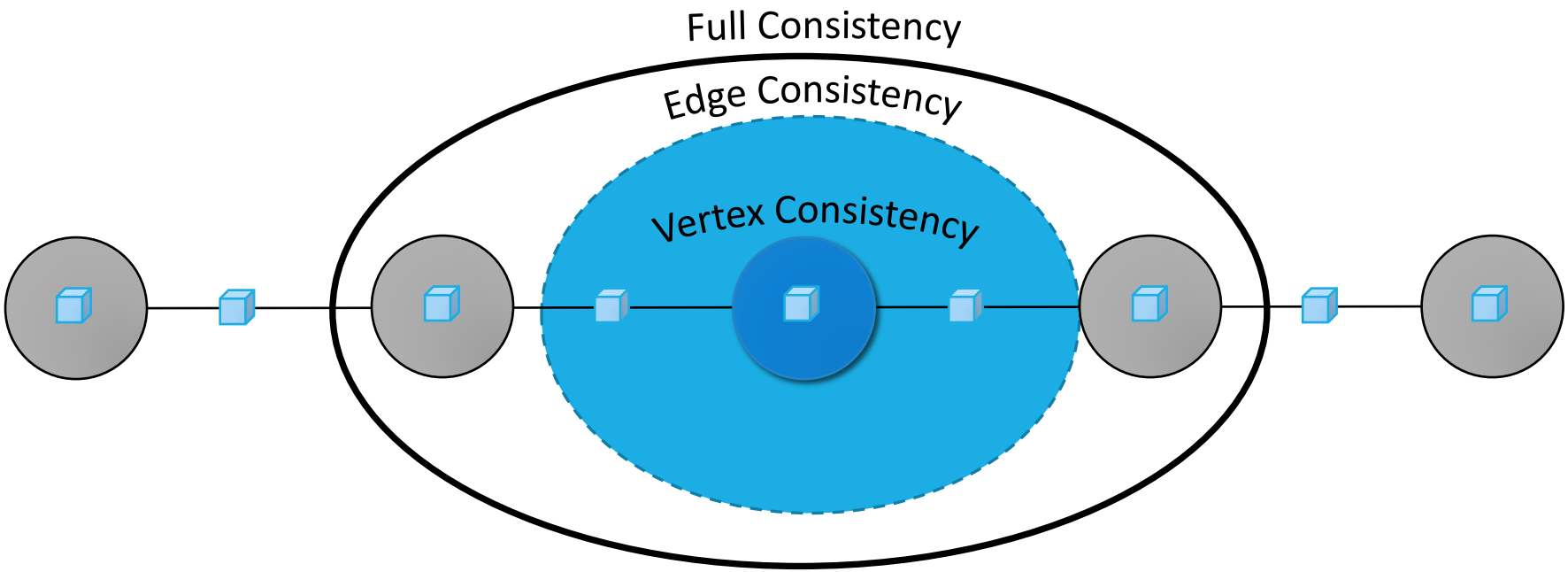
- e.g. simultaneous updates on the same vertex

Provides 3 levels of consistency

- Full consistency
- Edge consistency
- Vertex consistency

Each model guarantees the *scope* of vertices and edges that can be modified by the Update function

Data Consistency



Scheduling

A collection of basic schedulers

- Synchronous
- Round-robin

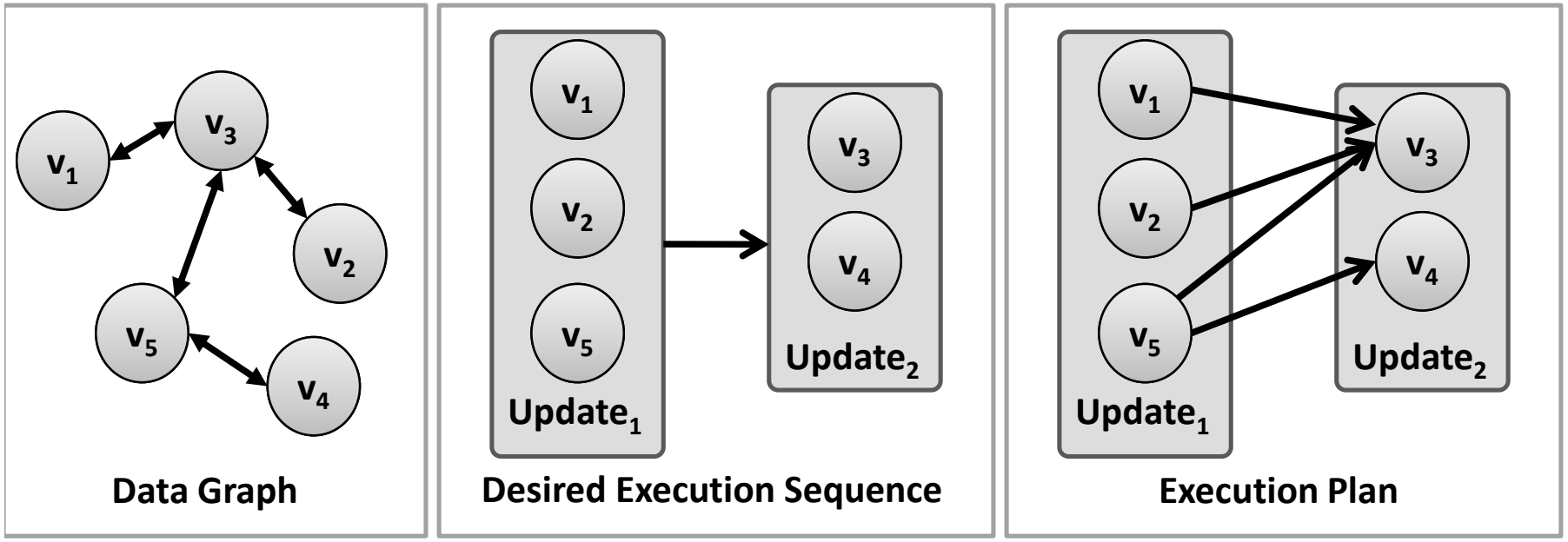
Tasks schedulers that allow task creation/reordering

- FIFO
- Prioritized

Provides users to create their own scheduler through a *Set Scheduler*

Set Scheduler

for $i = 1 \cdots k$ **do**
  Execute f_i on all vertices in S_i in parallel.
  Wait for all updates to complete

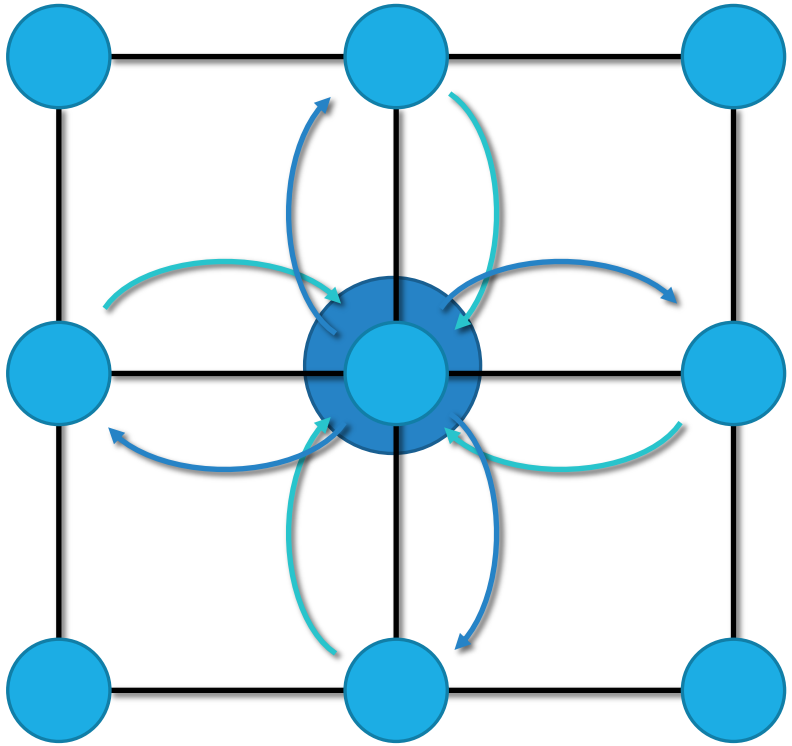


Case Study: Loopy Belief Propagation

Iteratively estimates ”beliefs” about vertices

- Read in messages
- Updates marginal estimate (belief)
- Send updated out messages

Repeat for all variables until convergence

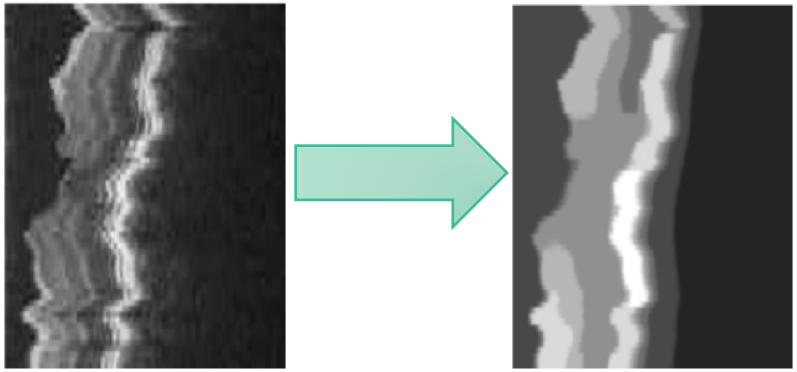


Case Study: Loopy Belief Propagation

Application: 3D retinal image denoising

- Represent as a 256x64x64 with each vertex as a voxel in the original image

3D retinal image denoising

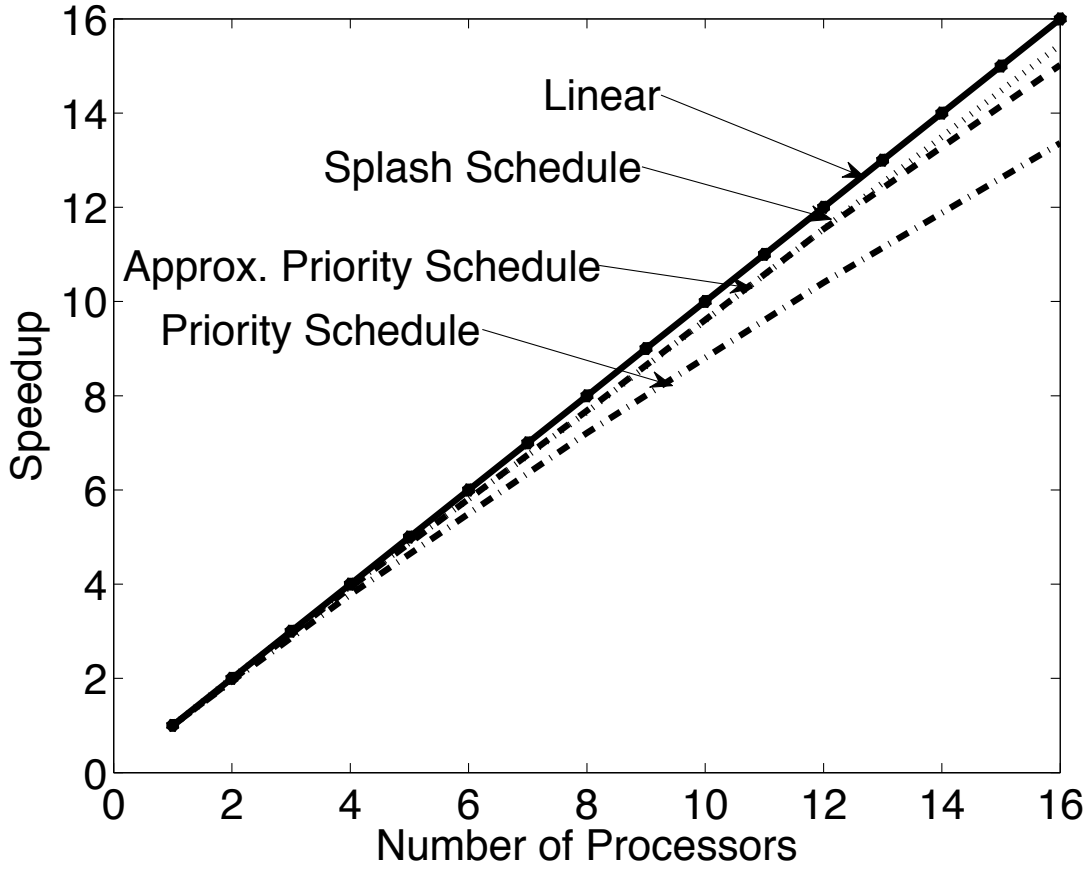


Case Study: Loopy Belief Propagation

Algorithm 2: BP update function

BPUpdate($D_v, D_{*\to v}, D_{v\to *} \in \mathcal{S}_v$) **begin**
    Compute the local belief $b(x_v)$ using $\{D_{*\to v} D_v\}$
    foreach $(v \to t) \in (v \to *)$ **do**
        Update $m_{v\to t}(x_t)$ using $\{D_{*\to v}, D_v\}$ and $\lambda_{\text{axis}(vt)}$ from the SDT.
        residual $\leftarrow \left|\left|m_{v\to t}(x_t) - m^{\text{old}}_{v\to t}(x_t)\right|\right|_1$
        if *residual* $>$ *Termination Bound* **then**
            AddTask(t, residual)
        end
    end
end

Case Studies: Loopy Belief Propagation



Case Studies

Other examples (Gibbs sampling, CO-EM, ...)

- Check the paper!

Limitations

Reports **self-scaling** numbers, but comparison with other frameworks or a serial baseline are missing.

Implicitly shared-memory model, how does it work for distributed systems?

Do ML algorithms really operate on large graphs?

Conclusion

Prior parallel frameworks unsuitable for ML

- High-level: not expressive enough
- Low-level: difficult to program

GraphLab provides a *vertex-centric* framework on data graphs

Scalability up to 16 cores on a wide range of ML applications