LLAMP: Assessing Network Latency Sensitivity and Tolerance of HPC Applications with Linear Programming

Siyuan Shen[1], Langwen Huang[1], Marcin Chrapek[1], Timo Schneider[1], Jai Dayal[2], Manisha Gajbe[2], Robert Wisniewski[3], Torsten Hoefler[1]

[1] ETH Zurich

[2] Samsung Semiconductor Inc.

[3] HPE

SC24 Atlanta, GA | hpc creates.

Memory and Communication Walls

Enabling 800G Ethernet Bandwidth for Hyperscale Data Centers

FEC Killed The Cut-Through Switch

Omer S. Sella   Andrew W. Moore   Noa Zilberman

It is crucial to analyze the effect of **growing network latency** on the performance of applications

Memory Access: Issues at Hyperscale

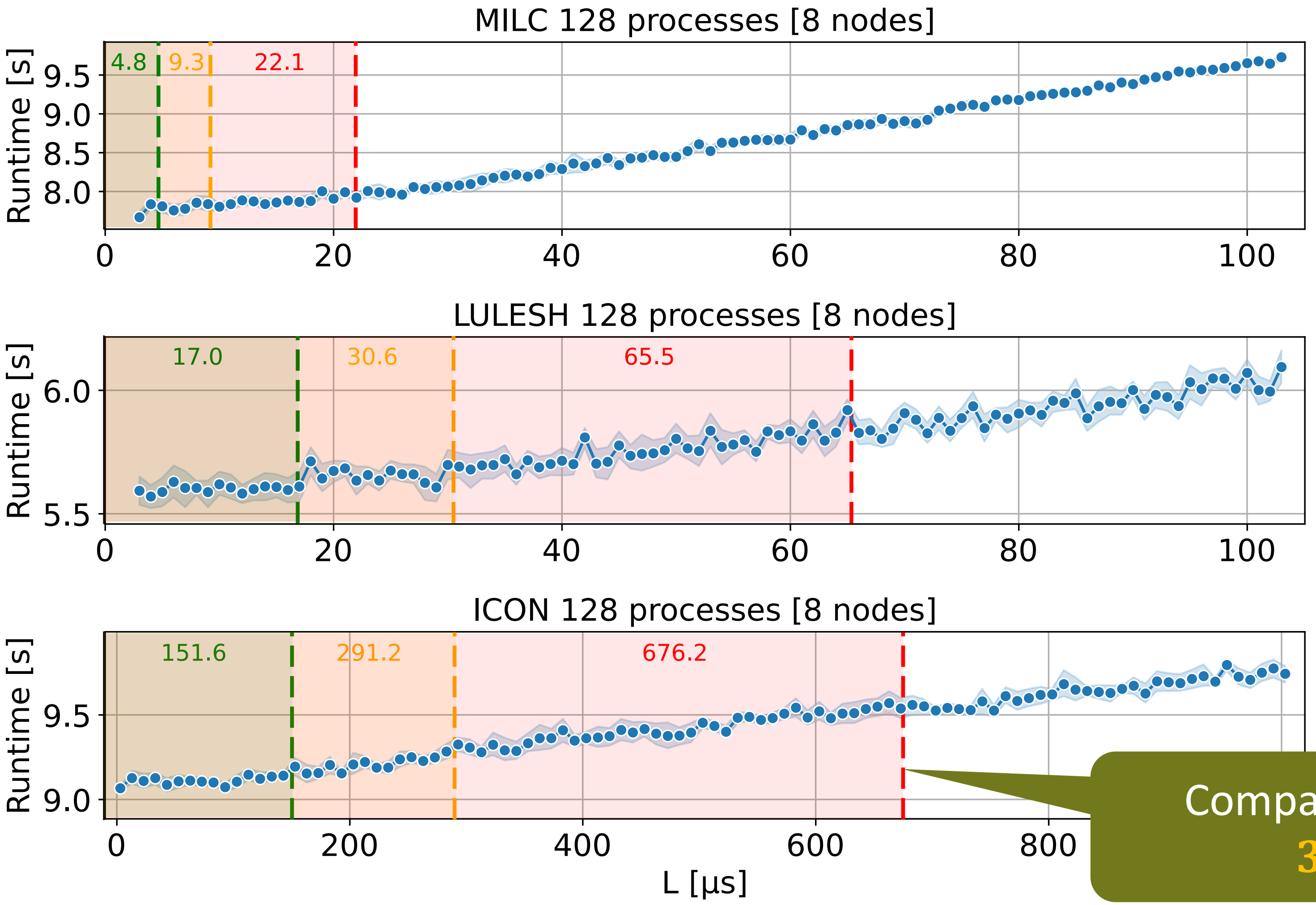
Torsten Hoefler, ETH Zürich
Duncan Roweth, Keith Underwood, and Robert Alverson, Hewlett Packard Enterprise
Mark Griswold, Vahid Tabatabaee, Mohan Kalkunte, and Surendra Anubolu, Broadcom
Siyuan Shen, ETH Zürich
Moray McLaren, Google
Abdul Kabbani and Steve Scott, Microsoft

The scaling of the bandwidth of different generations of interconnections & memory, as well as the peak FLOPS [1]

Higher network bandwidth leads to more complex **forward error correction (FEC)**, which will **increase latency**

[1] https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2022/11/ATPESC-2022-Track-1-Talk-3-Sury-Memory-Coupled-Compute.pdf

Communication Latency Tolerance of Different HPC Applications



The zones correspond to the maximum network latency before observing a performance degradation of 1%, 2%, and 5%

Network latency tolerance of HPC applications are different

Evaluate the effect of

Network topologies
e.g., Dragonfly vs. Fat Tree

Collective algorithms
e.g., Ring vs. Tree

Compared with MILC, ICON can absorb **30×** more network latency

Challenges of Measuring Communication Latency Sensitivity

Performance Modeling — ✗ Require in-depth understanding of applications' behaviors

Artificial Communication Latency Injection

Adding `sleep()` in software is not accurate!

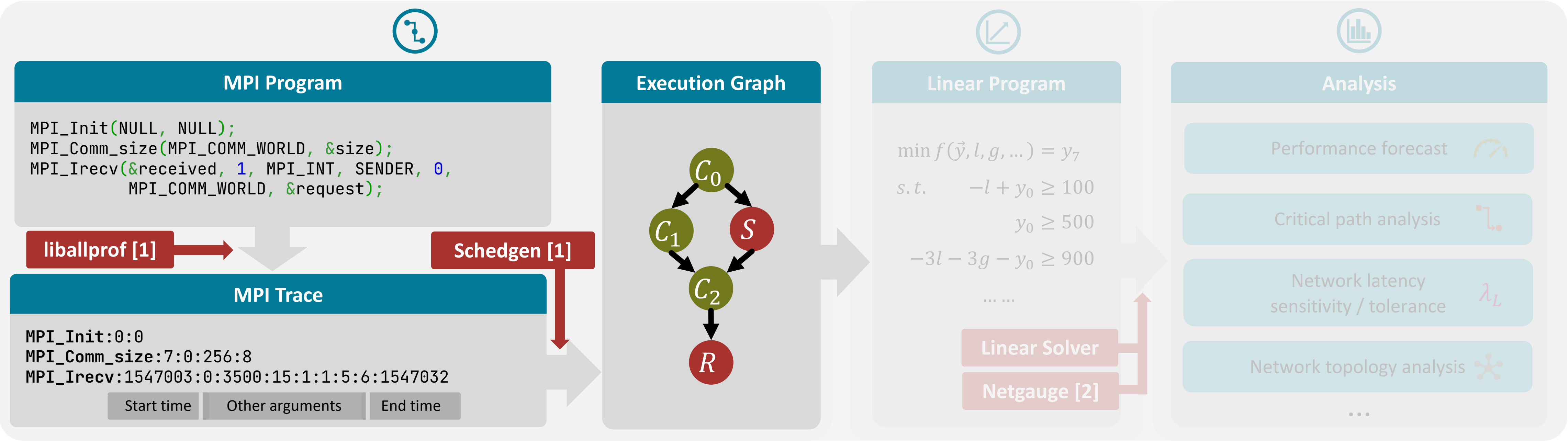
Hardware Modification — ✗ Difficult to procure and inflexible

Simulation — ✗ Execution is often very time-consuming

✗ Require extensive parameter sweeps (Hardware Modification and Simulation)

LLAMP Toolchain

LLAMP: LogGP and Linear Programming based Analyzer for MPI Programs



[1] Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. 2010. LogGOPSim: simulating large-scale applications in the LogGOPS model

[2] Torsten Hoefler, Torsten Mehlan, Andrew Lumsdaine, and Wolfgang Rehm. 2007. Netgauge: A Network Performance Measurement Framework

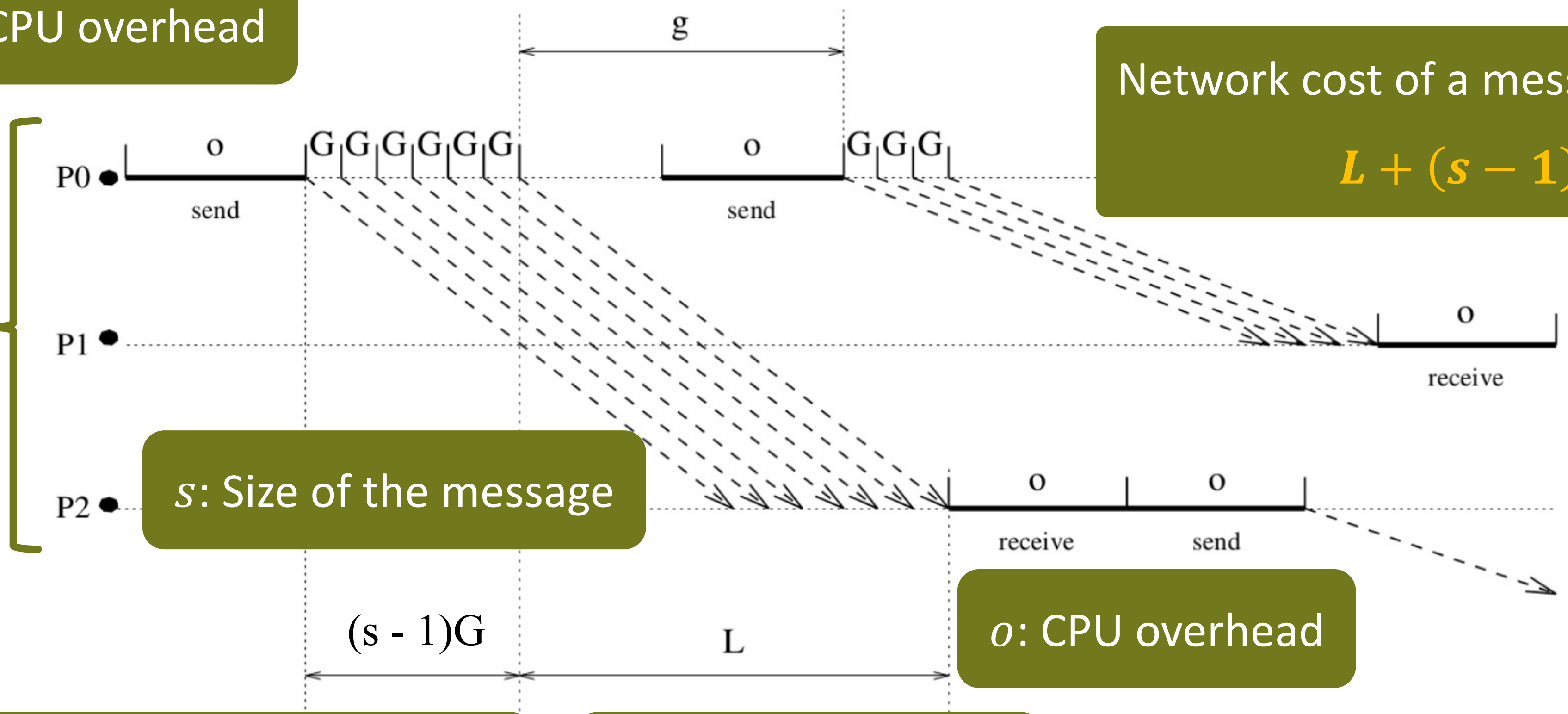
Background: The LogGP Model

o: CPU overhead

P: # of processes

Network cost of a message transfer:

$$L + (s-1)G$$



s: Size of the message

o: CPU overhead

G : Gap per byte (inverse bandwidth)

L: Network latency

Diagram taken from: Alexandrov, A., Ionescu, M. F., Schauser, K. E., and Scheiman, C. LogGP: incorporating long messages into the LogP model–one step closer towards a realistic model for parallel computation. ACM SPAA, July 1995.

Execution Graph Conversion

L: Network latency
o: CPU overhead
G: Gap per byte (inverse bandwidth)

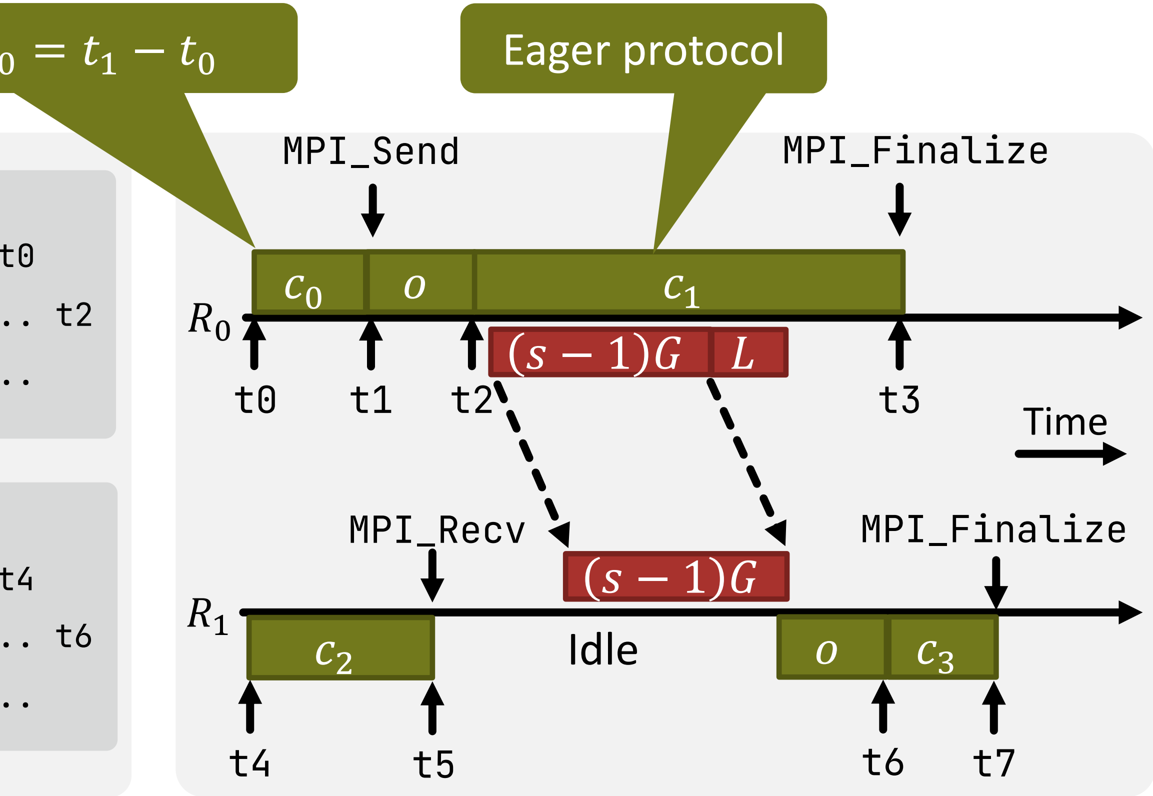
Rank 0

```
MPI_Init     : ... t0
MPI_Send     : t1 ... t2
MPI_Finalize: t3 ...
```

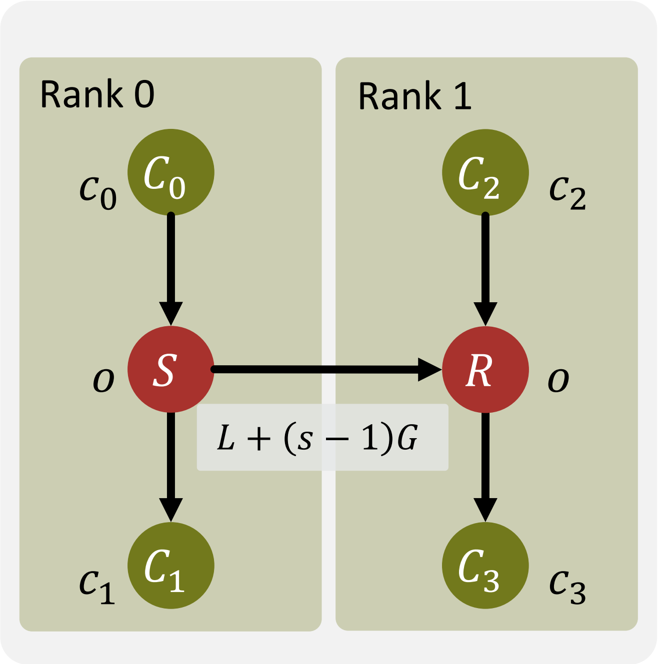
Rank 1

```
MPI_Init     : ... t4
MPI_Recv     : t5 ... t6
MPI_Finalize: t7 ...
```

Example MPI Traces



Space-time Diagram



Execution Graph

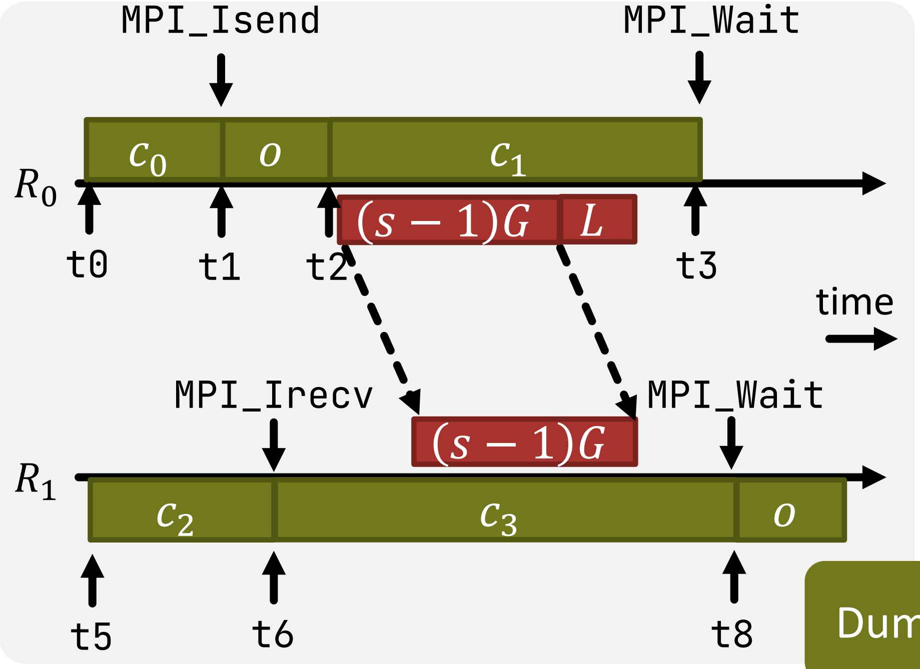
Execution Graph Conversion: Nonblocking Communication

Collectives are decomposed into individual sends and receives

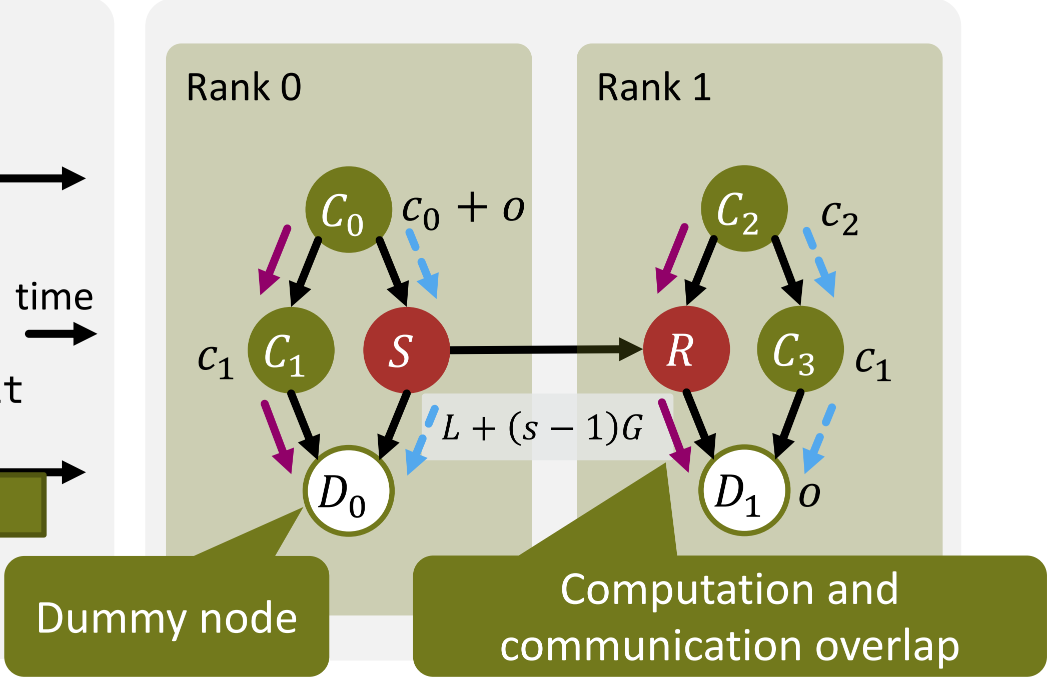
Rank 0
```
MPI_Init : ... t0
MPI_Isend: t1 ... t2
MPI_Wait : t3 ... t4
...
```

Rank 1
```
MPI_Init : ... t5
MPI_Irecv: t6 ... t7
MPI_Wait : t8 ... t9
...
```

Example MPI Traces



Space-time Diagram



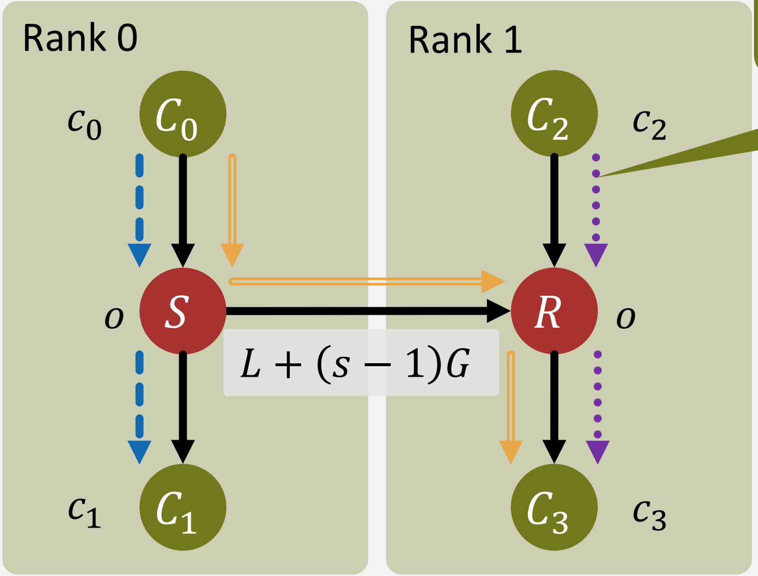
Execution Graph

Definition of Network Latency Sensitivity

Definition of ***network latency sensitivity***:

$$\lambda_L = \frac{\partial T}{\partial L}$$

Running Example



Three possible paths in this execution graph

$$T = \max\begin{pmatrix} c_0 + o + c_1, \\ c_0 + o + L + (s-1)G + c_3 + o, \\ c_2 + c_3 + o \end{pmatrix}$$

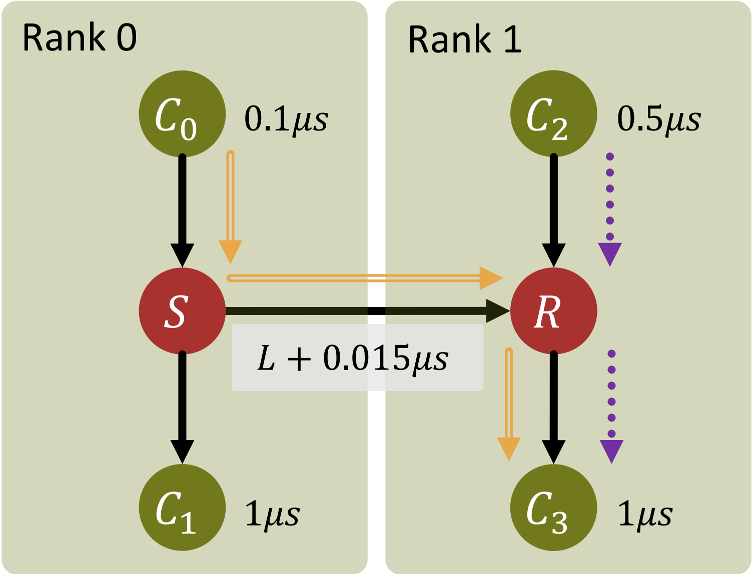
Execution time

Cost of each path

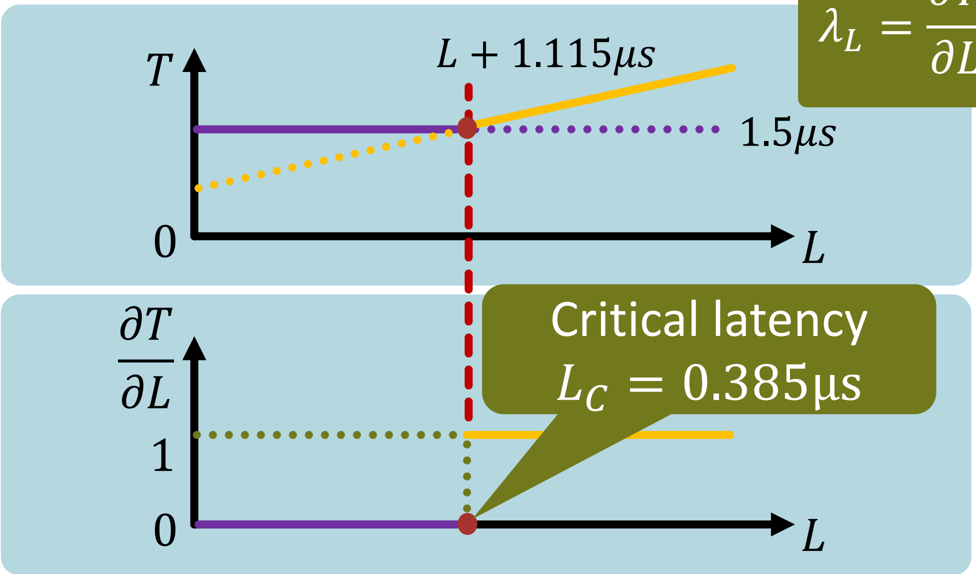
Definition of Network Latency Sensitivity: Example

$c_0 = 1\mu s$
$c_1 = 1\mu s$
$c_2 = 0.5\mu s$
$c_3 = 1\mu s$

$o = 0s$
$s = 4\ bytes$
$G = 5\ ns/byte$



$$\lambda_L = \frac{\partial T}{\partial L} = \begin{cases} 1, \text{if } L > 0.385\mu s \\ 0, \quad \text{otherwise} \end{cases}$$



$$T = \max\begin{pmatrix} c_0 + o + c_1, \\ c_0 + o + L + (s-1)G + c_3 + o, \\ c_2 + c_3 + o \end{pmatrix}$$

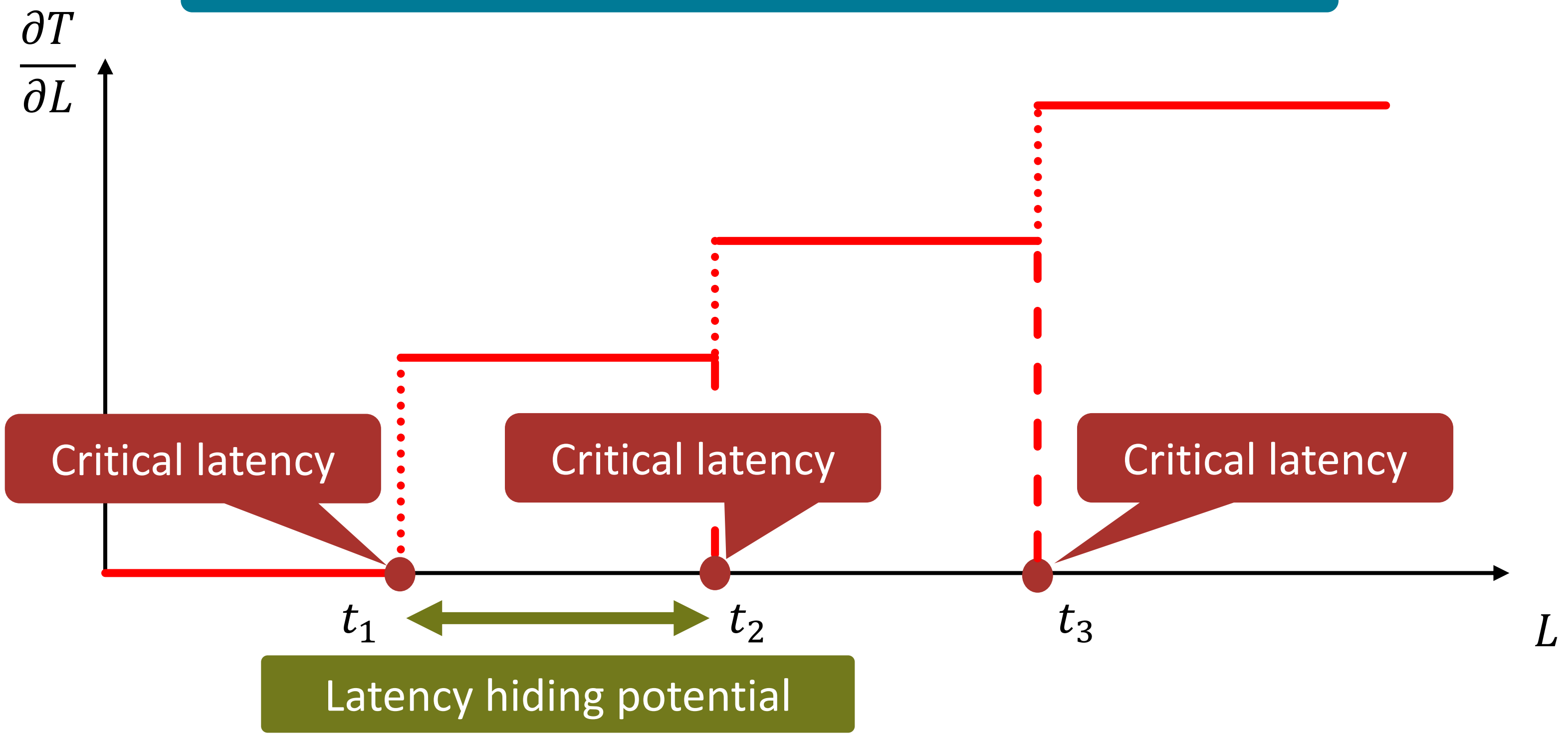
$$T = \max(L + 1.115\mu s, 1.5\mu s)$$

The **number of messages** along the critical path dictates $\frac{\partial T}{\partial L}$

Generalization: Latency Sensitivity Curve

To generalize, T is the maximum of the cost from all paths in the graph

The value of L has a **second-order effect** on $\frac{\partial T}{\partial L}$



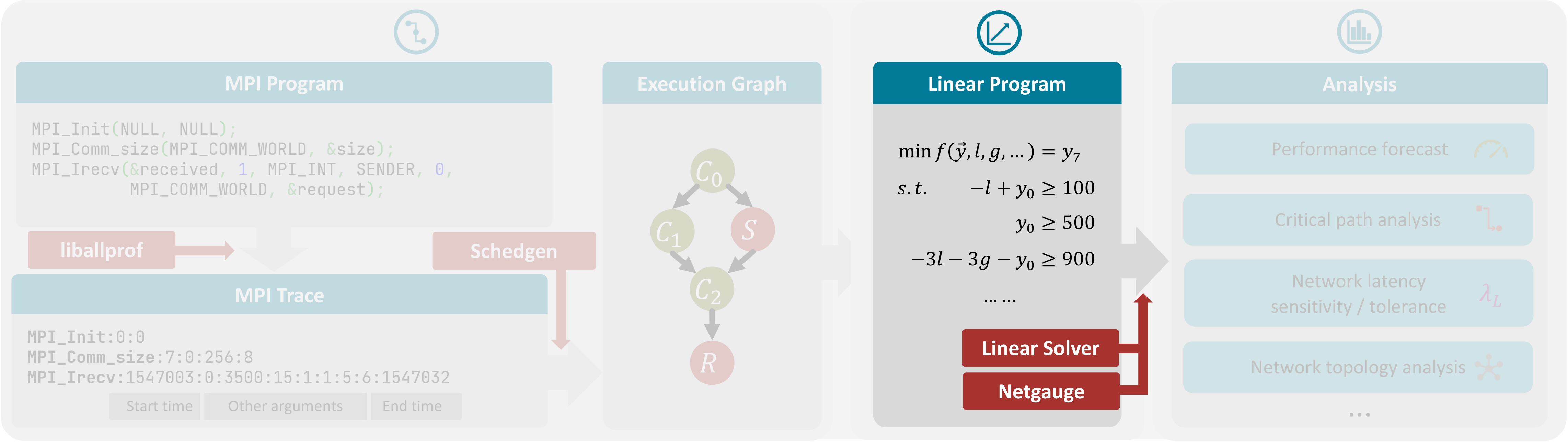
Calculate Network Latency Curve Analytically

Approach	Description	Verdict
Naïve	Find the cost for all paths in the graph	Intractable
Dynamic Programming	Space complexity: $O(n\|V\|)$ Time complexity: $O(n\|E\|)$	Not scalable
Linear Programming	Convert operation dependency graphs into **linear programs**	Solution

n: Length of the longest chain of communication

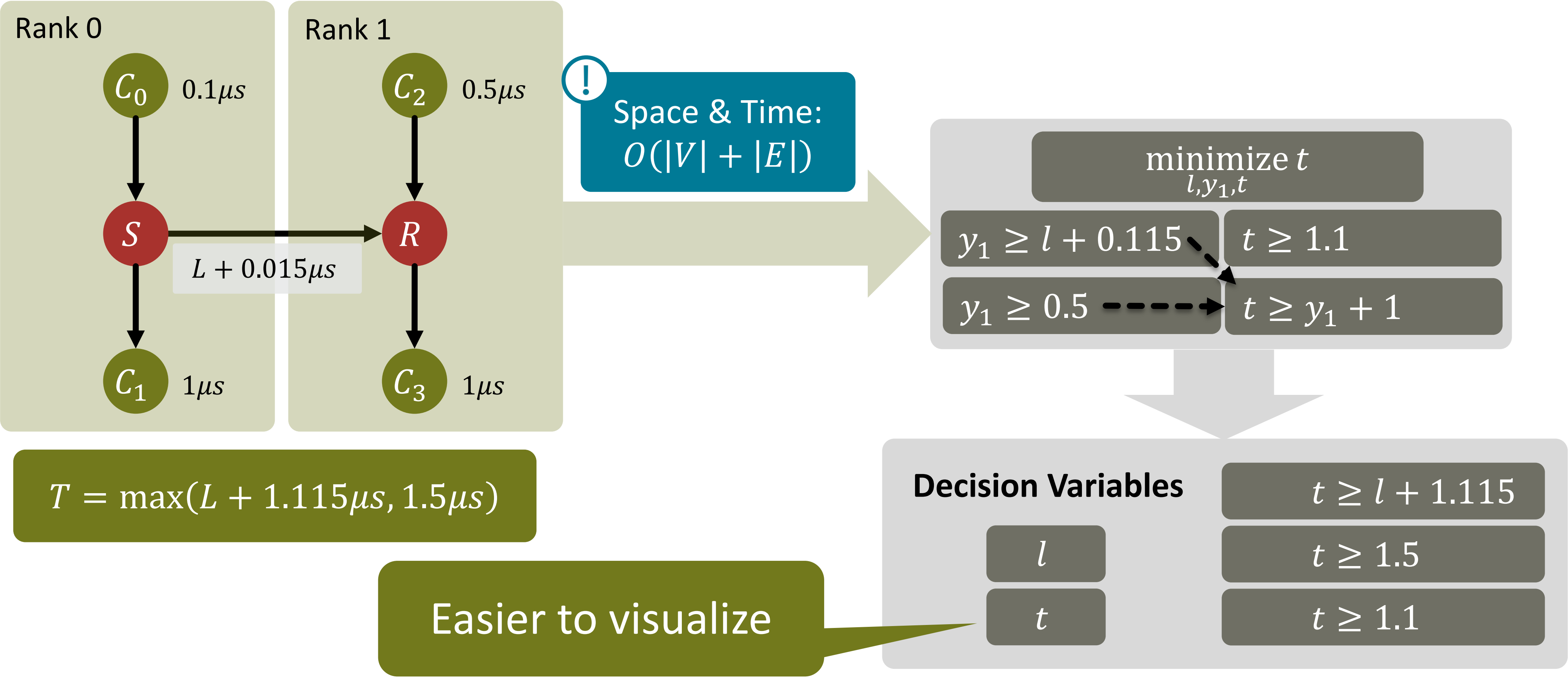
Graph of 470k vertices consumes > 50GB of memory

LLAMP Toolchain: Linear Programming (LP)



Convert Execution Graphs to Linear Programs (LPs)

LP



Connection between Graphs and Linear Programs

LP

$$\underset{l, y_1, t}{\text{minimize}}\ t$$

Constraints

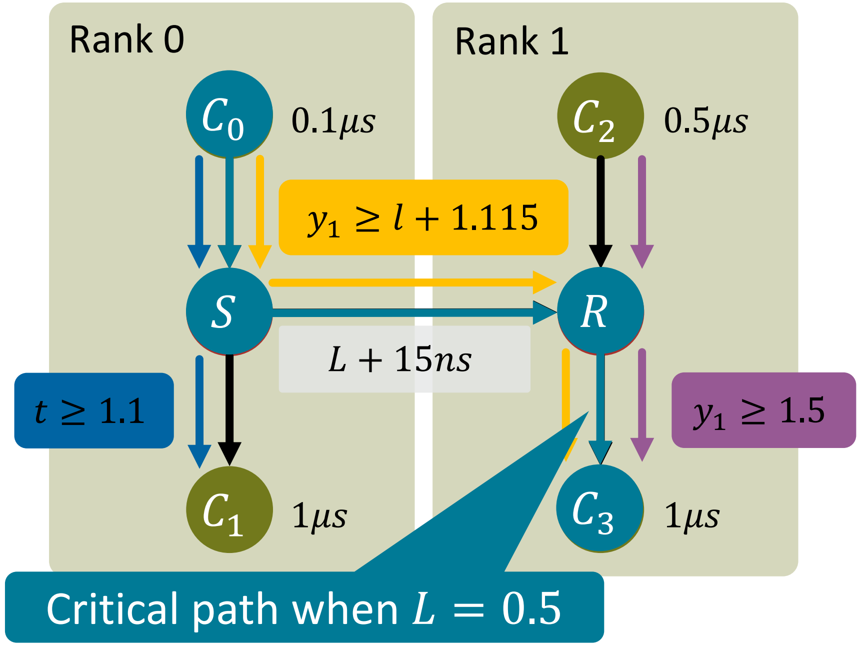
$t \geq l + 1.115$	$t \geq 1.5$
$t \geq 1.1$	$l \geq 0.5$

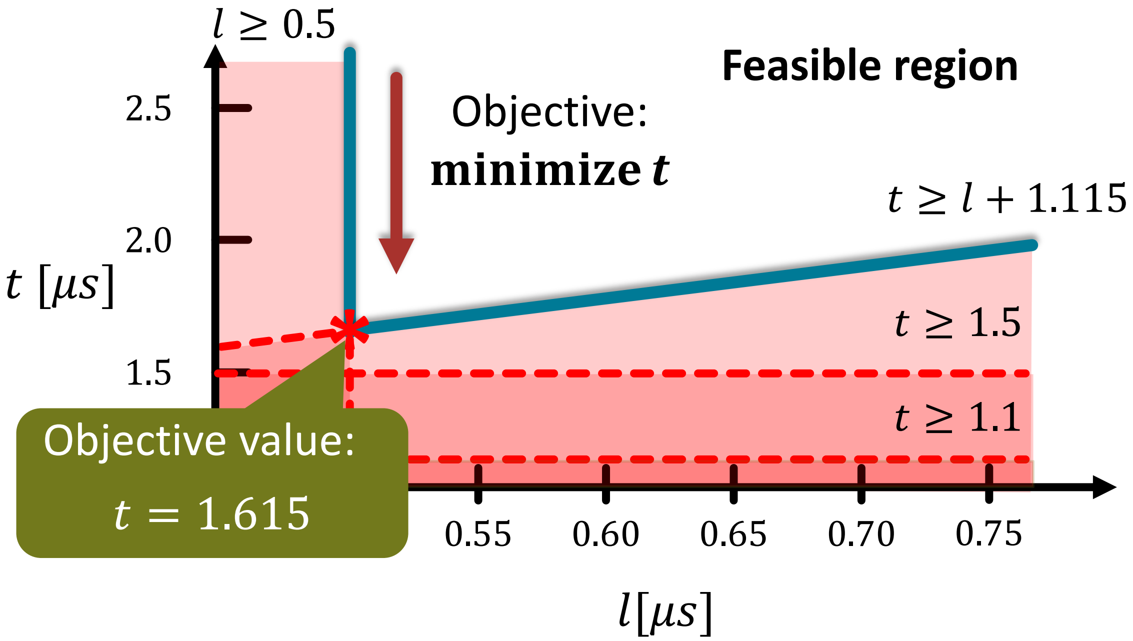
We aim to find the runtime when $L = 0.5\mu s$

! Constraints are representations of **path**

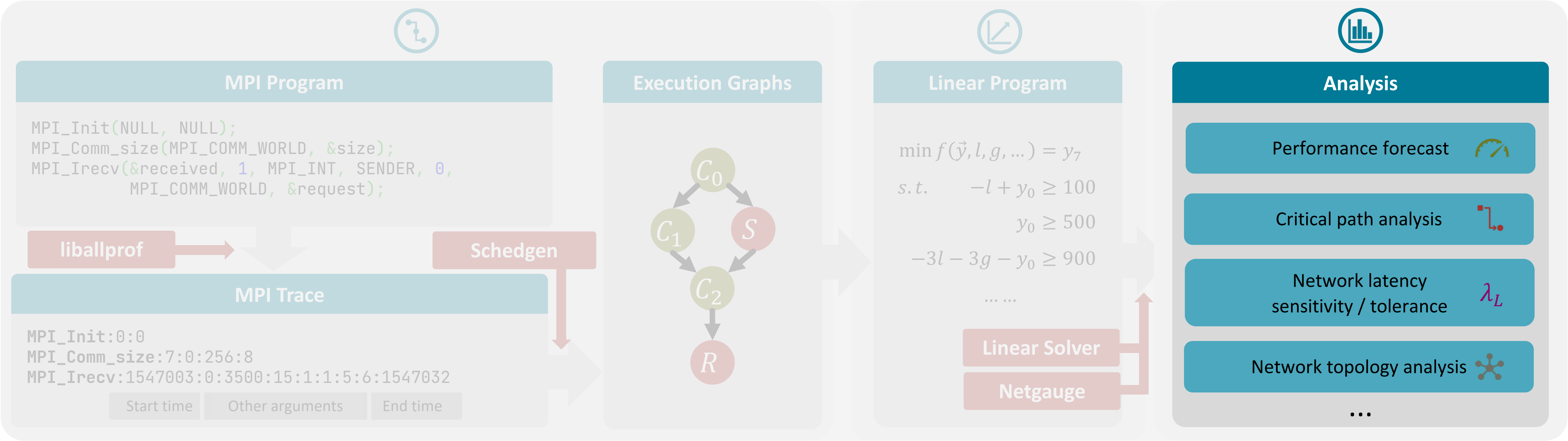
! Solving LPs finds the **critical path**

! LPs can be used for **runtime** prediction





LLAMP Toolchain: Analysis



Latency Sensitivity Curves

Linear Program

$$\underset{l, y_1, t}{\text{minimize}}\ t$$

Decision Variables

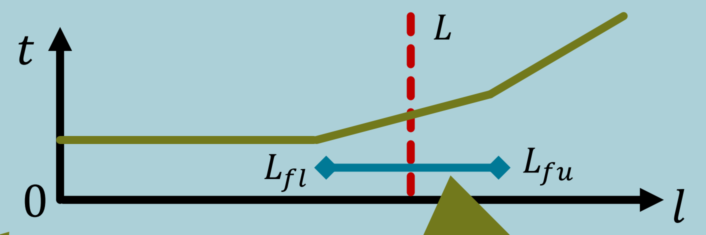
- l
- y_1
- t

Constraints

- $y_1 \geq l + 0.115$
- $y_1 \geq 0.5$
- $t \geq 1.1$
- $t \geq y_1 + 1$
- $l \geq L$

Network latency sensitivity, λ_L: ***number of messages*** on the critical path

Range of feasibility, $L_{fl} \leq l \leq L_{fu}$: an interval within which the critical path **remains the same**



No need to sweep across all L

$\frac{\partial T}{\partial L}$ is the same in $[L_{fl}, L_{fu}]$

Network Latency Tolerance

Analysis

Change the objective to **maximize l**

Linear Program

$\text{maximize}_{l,y_1,t}\ l$

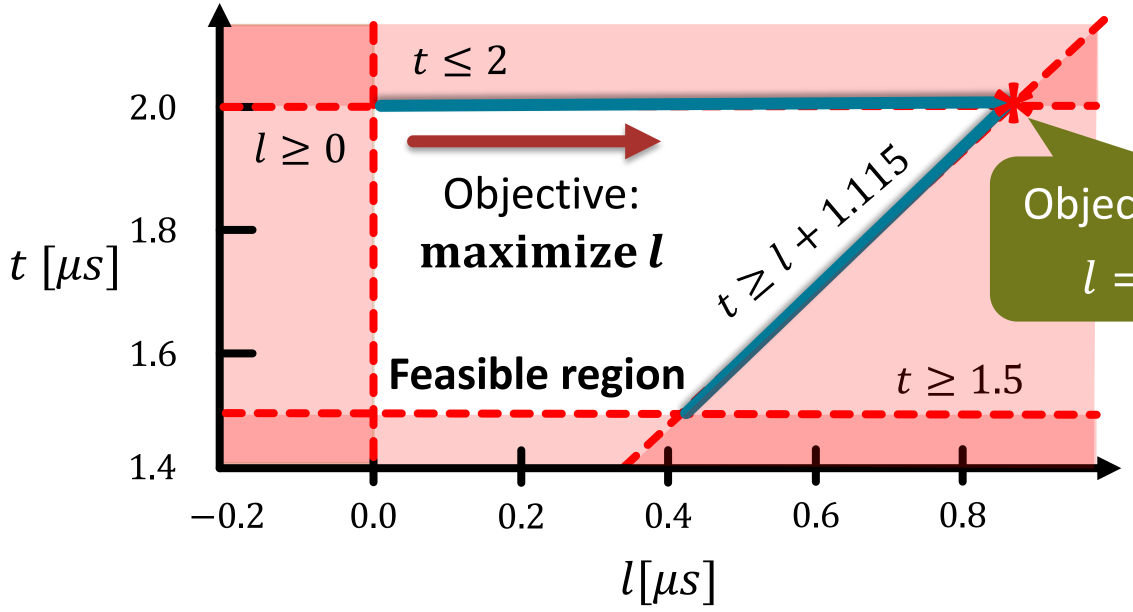
Decision Variables

- l
- y_1
- t

Constraints

- $t \geq l + 1.115$
- $t \geq 1.5$
- $t \geq 1.1$
- $l \geq 0$
- $t \leq 2$

Add a constraint to specify the maximum allowable execution time



Objective value: $l = 0.885$

Solving LPs yields the **maximum tolerable L** beyond which the runtime exceeds a **threshold** (e.g., 10% of baseline runtime)

Other Metrics

Bandwidth sensitivity:

Total number of bytes transferred along the critical path

Bandwidth tolerance:

Minimum bandwidth required to reach certain performance

minimize g

LLAMP is extremely versatile!

$$\rho_L = \frac{\lambda_L \cdot L}{T}$$

$$\rho_G = \frac{\lambda_G \cdot G}{T}$$

Heterogenous LogGP:

Support for heterogenous network by using different decision variables to represent different L.

Topology analysis:

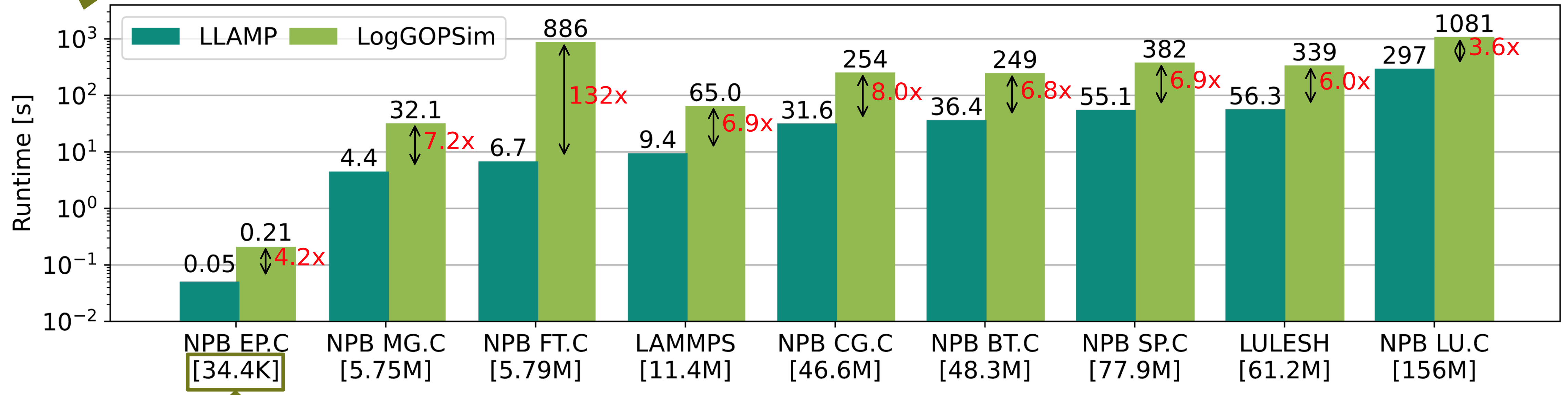
Support for analyzing network topologies by replacing the latency of individual wires with a decision variable.

How Fast is Linear Programming?

LogGOPSim [1] is one of the most efficient open-source network simulators

LLAMP uses Gurobi linear solver

Logscale



Number of events in the execution graph

! Linear solvers can remove redundant constraints during **presolve**

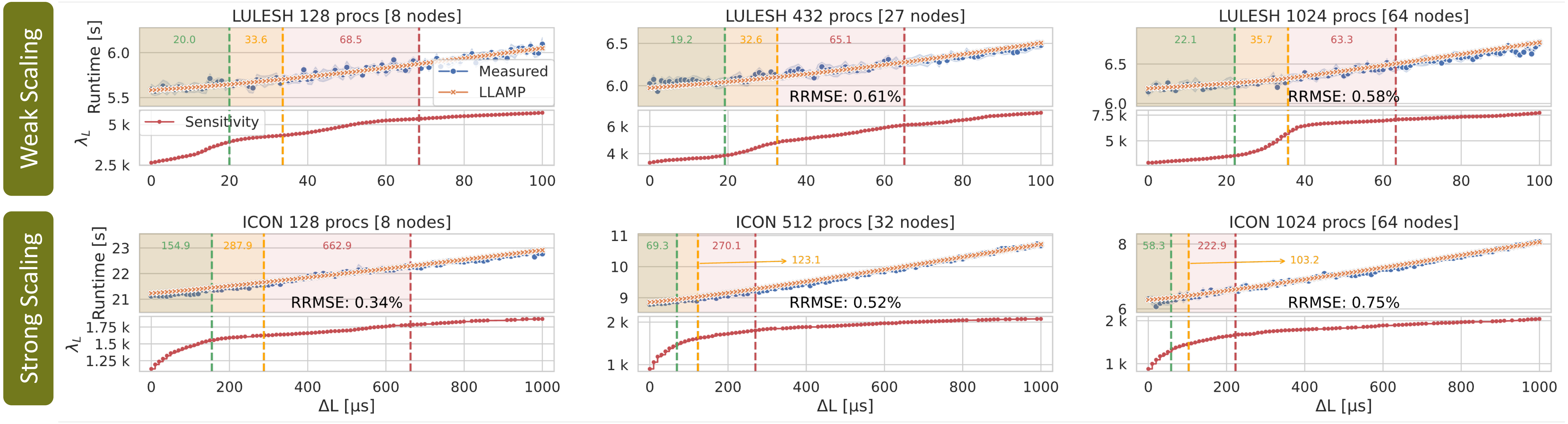
[1] Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. 2010. LogGOPSim: simulating large-scale applications in the LogGOPS model

Validation

We measured runtimes of 7 applications and compared them with the predictions from LLAMP

Real system: ~45 hours
LogGOPSim: ~14 hours
LLAMP: **~5 hours** (including tracing and preprocessing)

The relative root mean square errors (RRMSE) is below **2%** for all applications

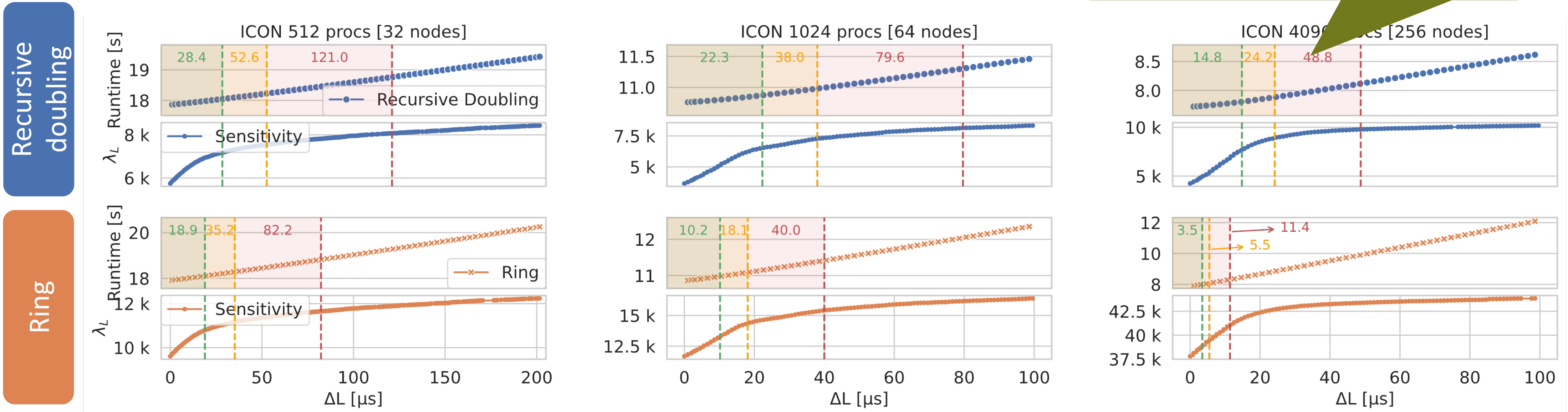


ICON Case Study: Collective Algorithm

We compare the performance impact of two allreduce algorithms: **recursive doubling** and **ring**

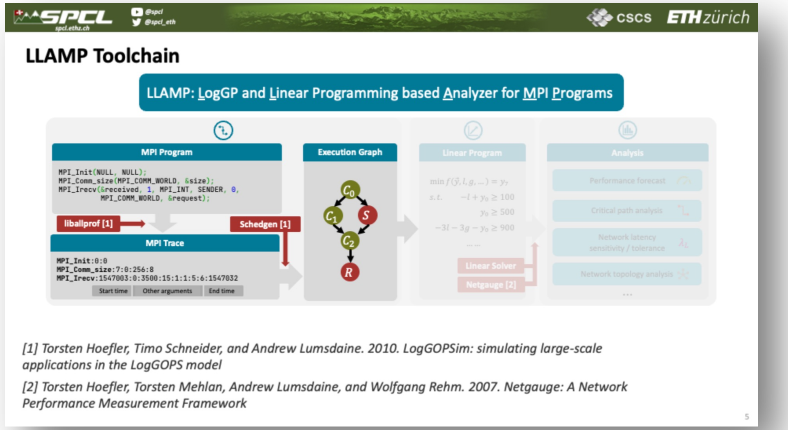
ICON's performance becomes more sensitive to L when the ring algorithm is used

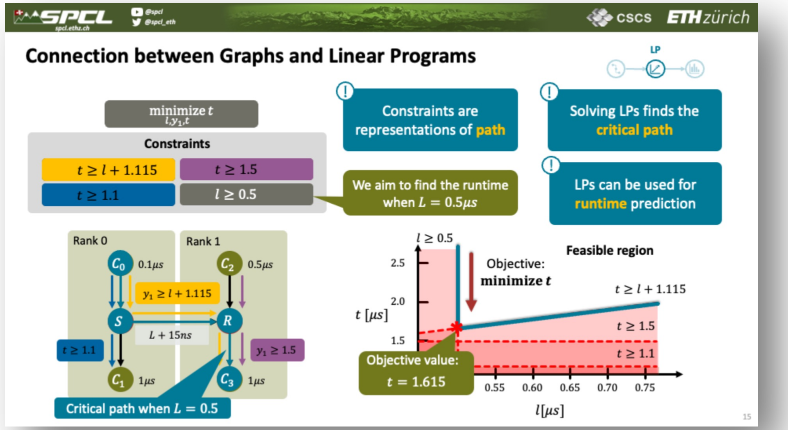
~4× network latency tolerance

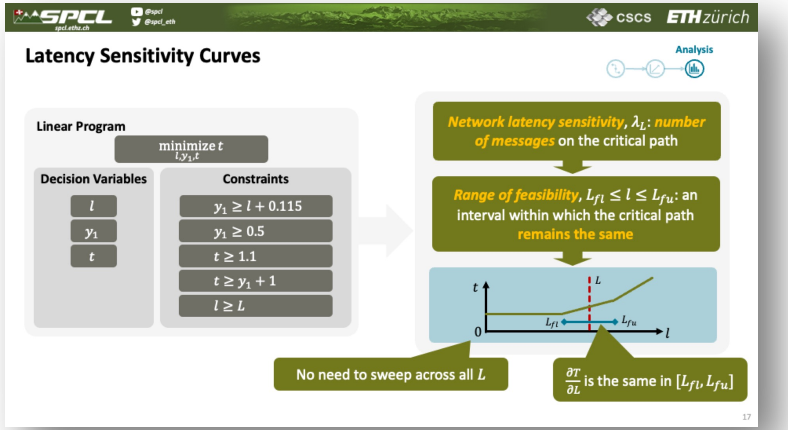


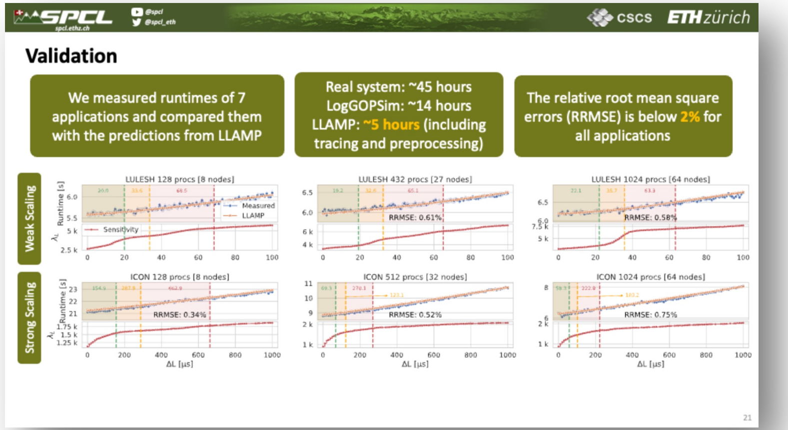
LLAMP can be used for designing **collective algorithms**

Conclusions









More of SPCL's research:

- youtube.com/@spcl — 210+ Talks
- twitter.com/spcl_eth — 1.6K+ Followers
- github.com/spcl — 5.6K+ Stars

... or spcl.ethz.ch

More results in the paper:

https://arxiv.org/abs/2404.14193

https://github.com/spcl/llamp

Backup Slides

Visualization of Linear Program

Linear Program

$$\underset{l, y_1, t}{\text{minimize}}\ t$$

Decision Variables

- l
- y_1
- t

For easier visualization, we integrate the value of y_1 into $t \geq y_1 + 1$

Constraints

- $y_1 \geq l + 0.115$
- $y_1 \geq 0.5$
- $t \geq 1.1$
- $t \geq y_1 + 1$
- $l \geq L$

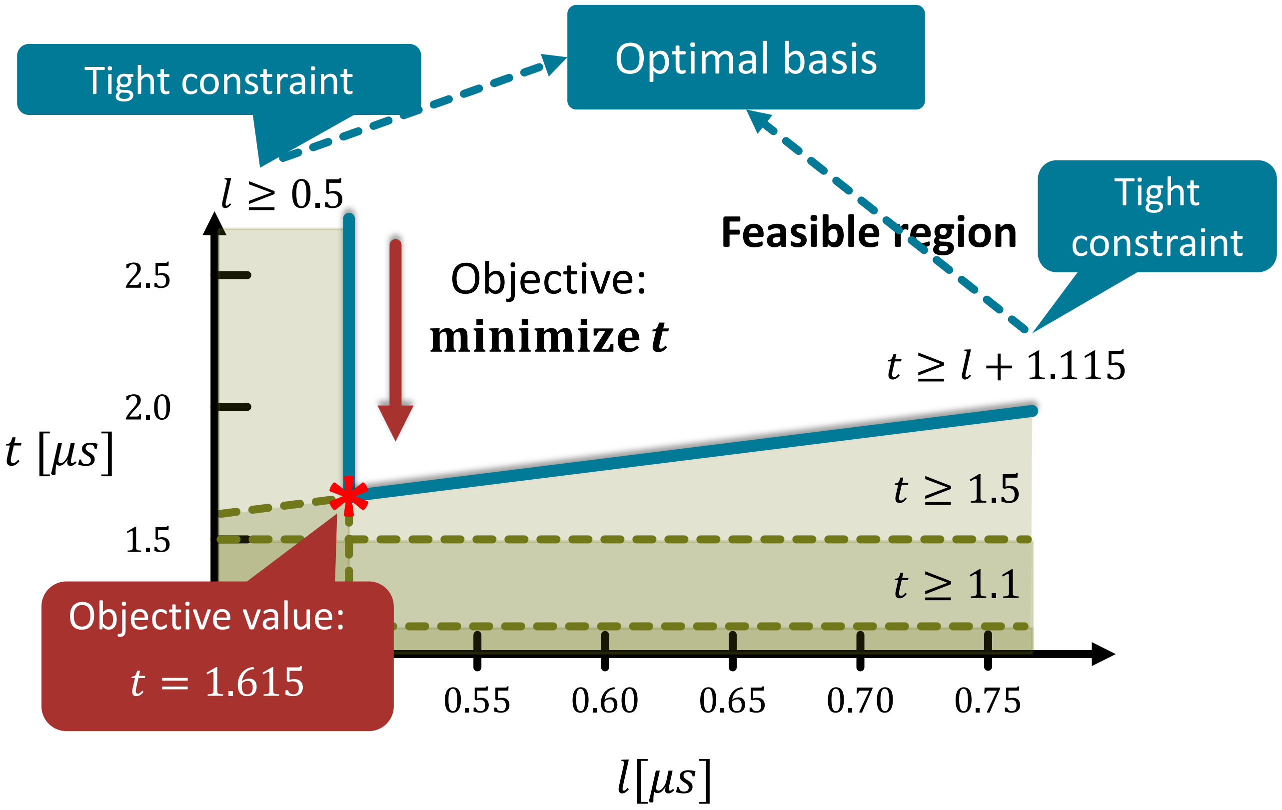
Constraints

- $t \geq l + 1.115$
- $t \geq 1.5$
- $t \geq 1.1$
- $l \geq 0.5$

We aim to compute the runtime when $L \geq 0.5\mu s$

$C_0 \rightarrow S \rightarrow C_1$ $C_0 \rightarrow S \rightarrow R$ $C_2 \rightarrow R$ C_3

$$T = \max(1.1, \max(L + 0.115, 0.5) + 1)$$



! A constraint is *tight* if it defines a boundary of the polyhedron where the optimum lies.

! A set of tight constraints represent the *optimal basis*.

Construct Linear Programs from Mathematical Expression

$C_0 \to S \to C_1$ $C_0 \to S \to R$ $C_2 \to R$ C_3

$$T = \max(1.1, \max(L + 0.115, 0.5) + 1)$$

Linear Program

$$\underset{l,y_1,t}{\text{minimize}}\ t$$

Decision Variables	Constraints
l	$y_1 \geq l + 0.115$
y_1	$y_1 \geq 0.5$
t	$t \geq 1.1$
	$t \geq y_1 + 1$
	$l \geq L$

① Create a decision variable $\boldsymbol{l}$ to represent network latency.

② Turn each *max* operator into a new decision variable and two constraints.

When the outermost *max* is encountered, use decision variable $\boldsymbol{t}$ to represent time.

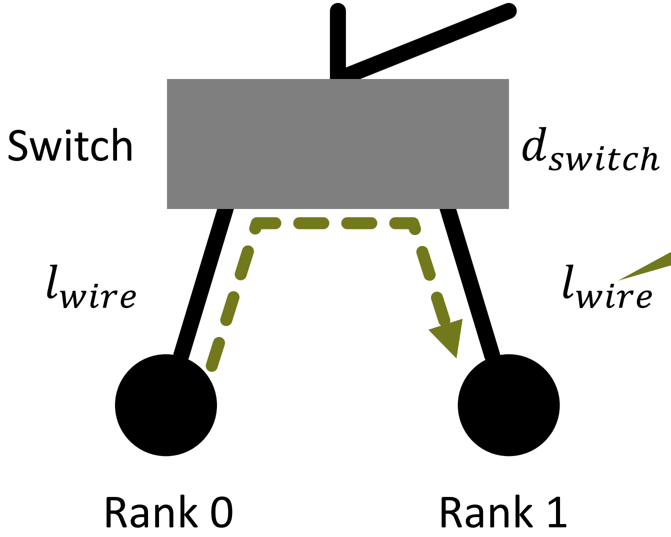
③ Set the objective of the linear program to **minimize $\boldsymbol{t}$**

④ Add the constraint $l \geq L$ and solve for the objective value to obtain T for a given L.

! Space complexity: $O(|V| + |E|)$

time complexity: $O(|V| + |E|)$

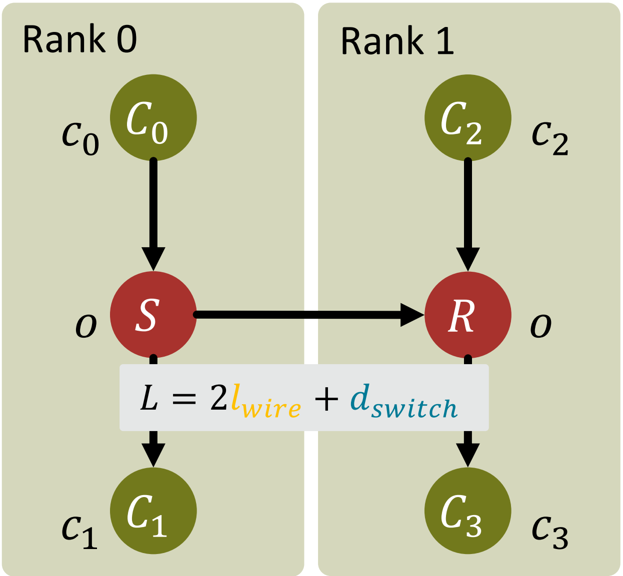
ICON Case Study: Topology

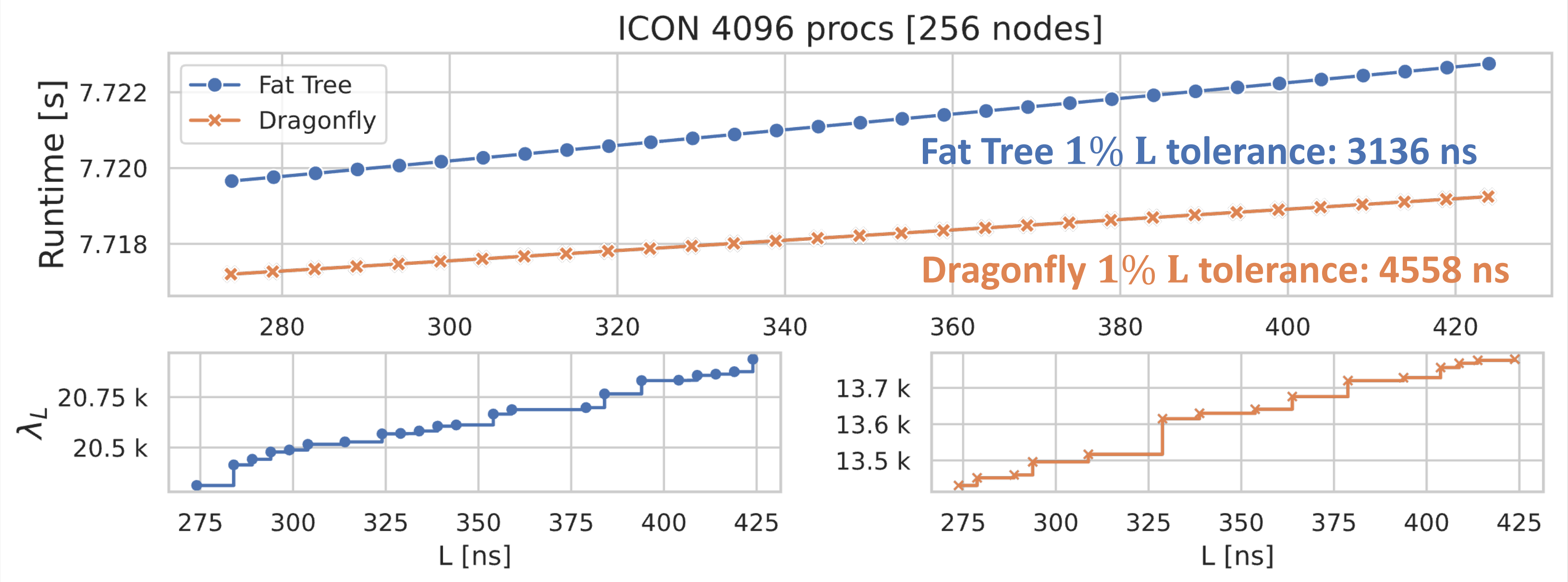


Substitute the latency of all wires with a decision variable l_{wire}.

Transmission latency: $(h+1) \cdot l_{wire} + h \cdot d_{switch}$

h: number of hops between the nodes





Future Work

A generalized strategy for analyzing different parallel programming models (e.g., charm++, Legion)

Communication sensitivity analysis for machine learning training and inference