Extensible Simulator for Replay of Trace Files in the Pajé Format

Tais Loureiro Bellini

Advised by Prof. Dr. Lucas Mello Schnorr

Bachelor Dissertation Presentation
Porto Alegre, May 30th 2016

Outline

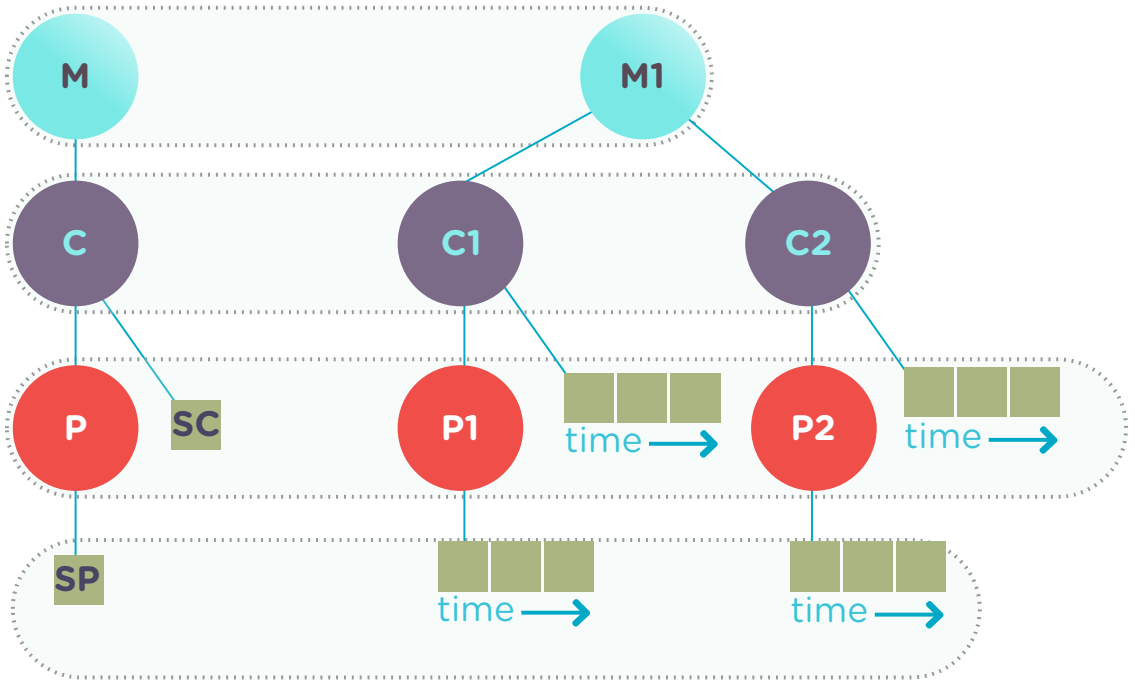
Context

- Trace files in the context of Performance Analysis
  - Important events in program executions are logged
- Replay of trace files
  - Richer entities (combination of multiple events)
  - Analyze program behavior $\rightarrow$ Optimizations

- Pajé Trace File Format
- Pajé Visualization Tool and PajeNG

Pajé Trace File Format

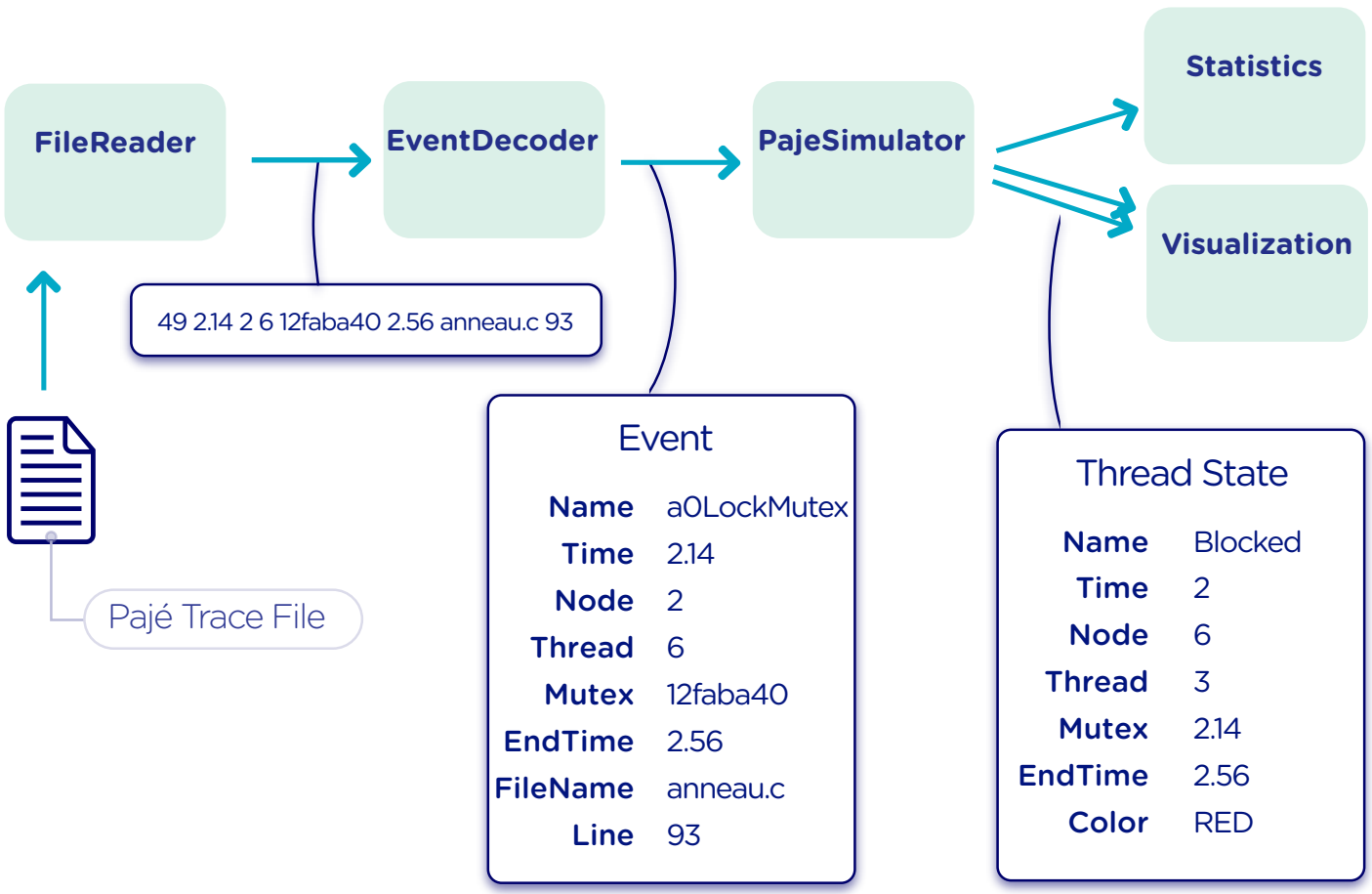
- Entities: Container, State, Link, Variable and Event
- Types and entities hierarchy



- Header and body sections

PajeNG Architecture

- Designed to support extensions in Pajé Format
- `pj_dump` tool



PajeNG Issues

- Extensibility
  - little flexibility in the manipulation of data
- Partial results
  - full set of results dumped at once
- Permanent results
  - results are discarded at the end

PajeNG Issues

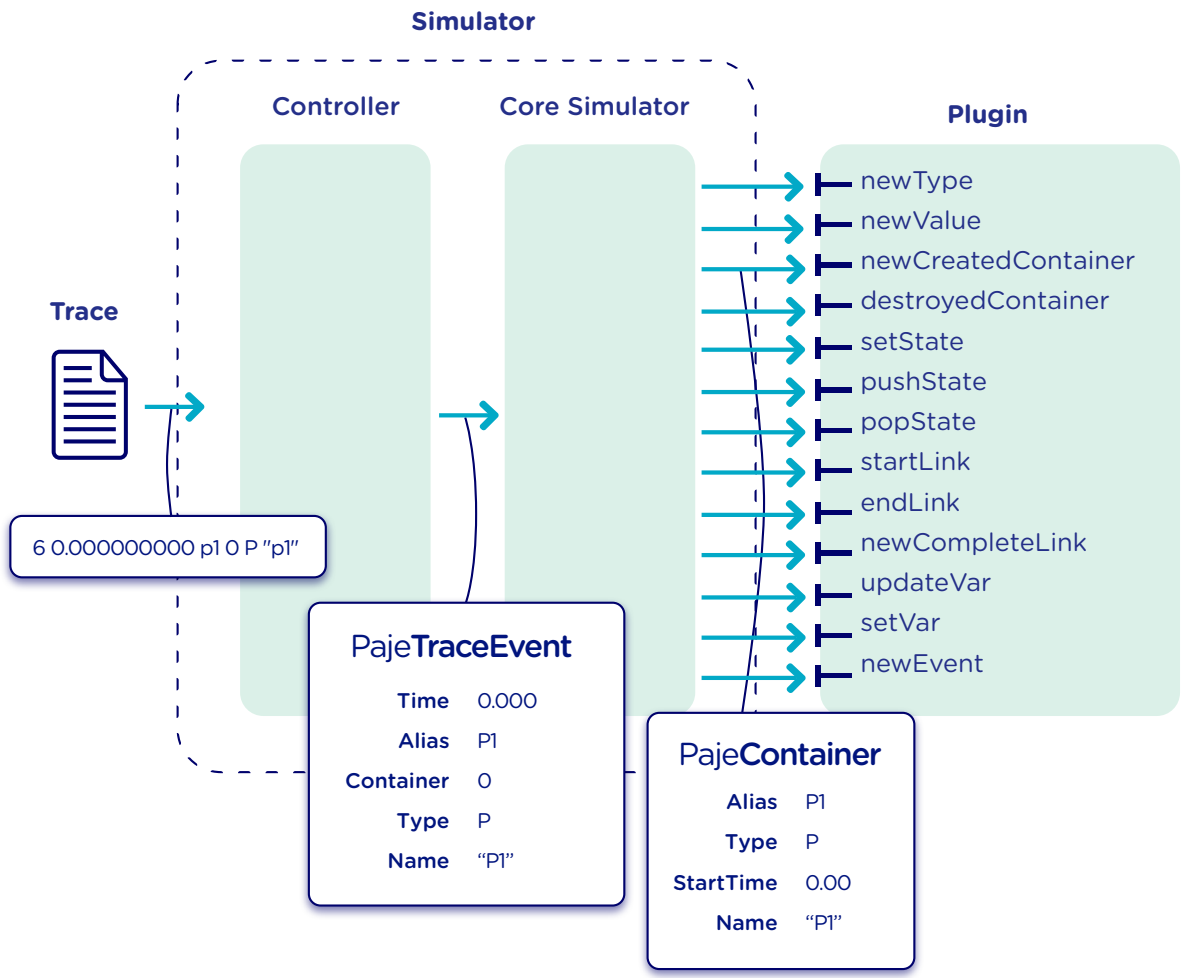
- Extensibility
  - little flexibility in the manipulation of data
- Partial results
  - full set of results dumped at once
- Permanent results
  - results are discarded at the end

Objective of our work

- Solve these issues by proposing a new extensible simulator that makes the core simulation transparent to the user facilitating the manipulation of data

Aiyra Architecture

- Three packages: controller, core simulator and plugin
  - Every event detected goes through all packages
- Keep memory footprint as low as possible



Plugins

Three plugins to validate the proposal

- PajeNullPlugin
  - No treatment for entities
- PajeDumpPlugin
  - Dump entities to standard output
- PajeInsertDBPlugin
  - Insert entities in a relational database
  - JDBC and MySQL
  - Insertion using JDBC Batches

Experimental Evaluation Methodology

- Full Factorial and Randomized Experimental Design
- Experiments:
  1. Comparison between Aiyra and PajeNG
  2. Different batch sizes in the PajeInsertDBPlugin

	Luiza	**Orion1**	**Guarani**
Processor	Core i7	Xeon E5-2630	Core i5-2400
CPU(s)	1	2	1
Cores per CPU	4	6	4
Max. Freq.	2.7 GHz	2.30GHz	3.10GHz
L1d/L1i Cache	32/32KBytes	32/32KBytes	32/32KBytes
L2 Cache	256KBytes	256KBytes	256KBytes
L3 Cache	6MBytes	15MBytes	6MBytes
Memory	16GBytes	32GBytes	20GBytes
OS	OSX 10.10.5	Ubuntu 12.04.5	Debian 4.3.5-1

1. The Aiyra against PajeNG Case

Setup: Aiyra with PajeNullPlugin, PajeNG without dump
Goal: Evaluate the difference in the execution time

Factors and Levels

- Platform: luiza, orion1, guarani
- Version: aiyra, pj, pjflex
- Input: big(1G), medium(128M), small(128K)

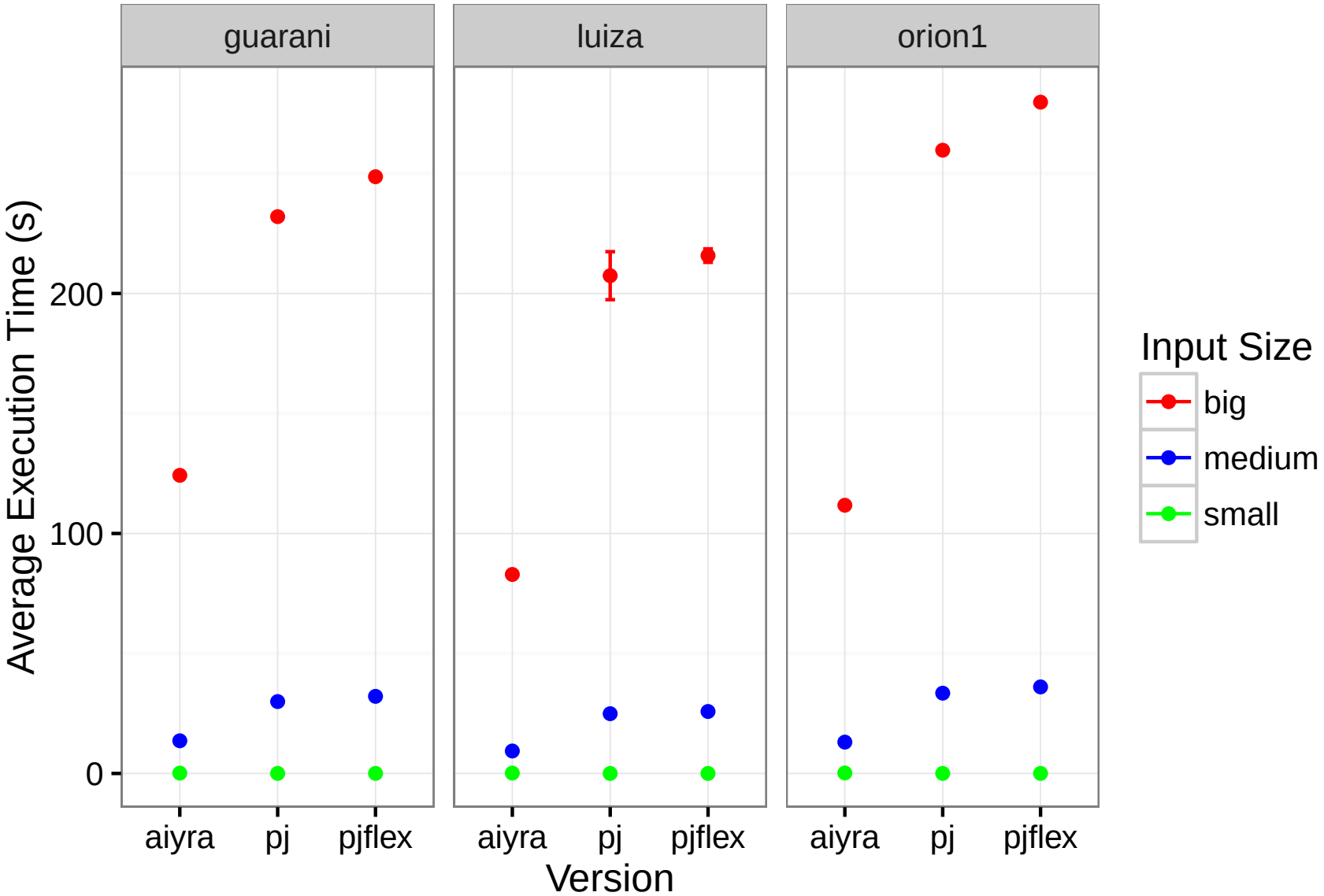
Response variable $\rightarrow$ Execution time
Statistical framework

- Average of 30 runs – CI of 99.7% (assuming gaussian)
  - SE: $3 \times sd/sqrt(N)$, where N is the number of replications

Expected behavior: PajeNG (pj, pjflex) faster than Aiyra

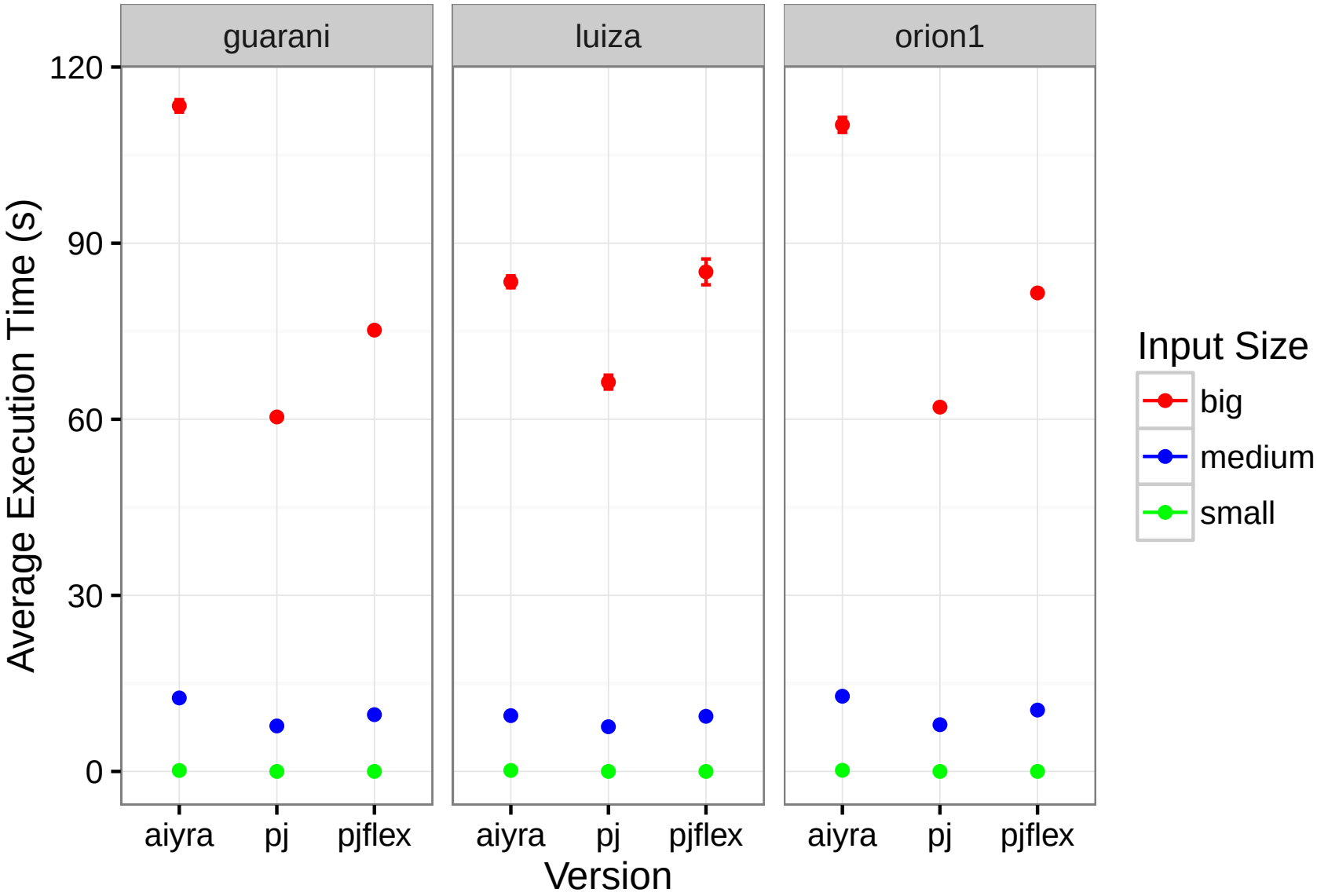
Results of Aiyra against PajeNG (unoptimized)

Comparison between Aiyra and PajeNG without optimization flags



Results of Aiyra against PajeNG (optimized)

Comparison between Aiyra and PajeNG with compilation flag -O3



2. The Batch Size Evaluation (InsertDBPlugin)

Setup: Aiyra with InsertDBPlugin
Goal: What is the best batch size?

Factors and Levels

- Input: big, medium, small
- Batch Sizes
  - Luiza (A, B, C, D, E, F) Guarani and Orion1 (B, C, D, E, F)

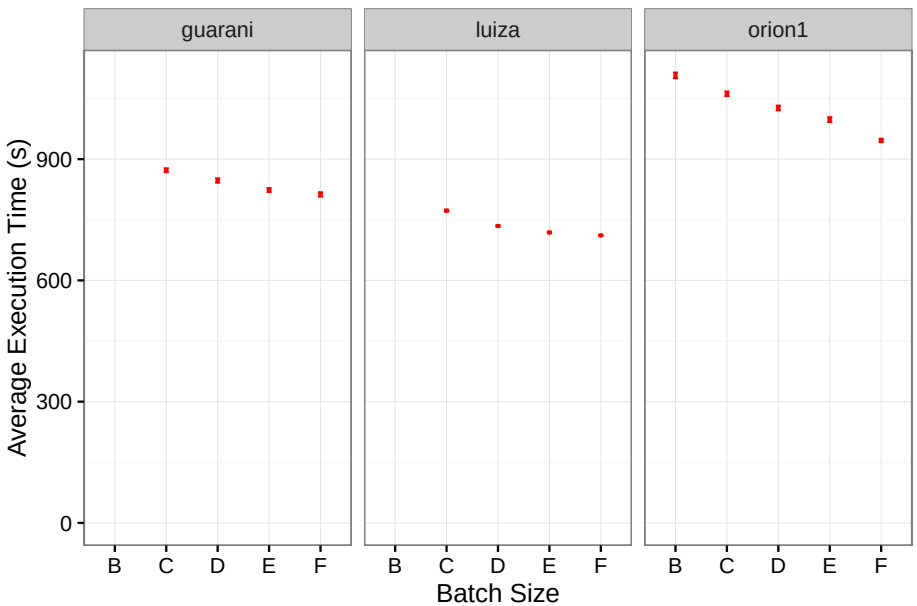
Response Variables → Execution time, Maximum memory usage, Insertion time, Batch insertion traces (temporal series)

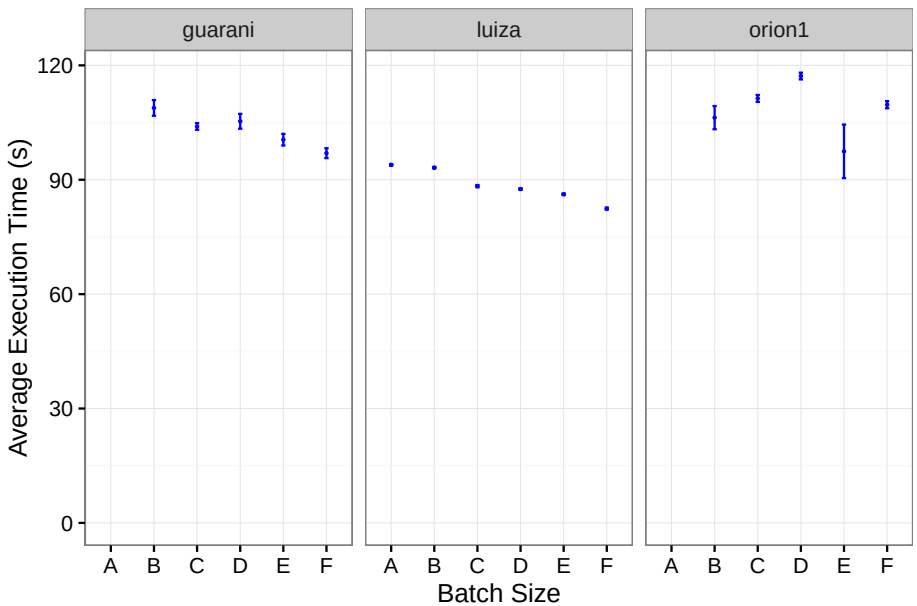
Replications → Luiza: 30 Guarani and Orion1: 10
JVM Maximum Heap Sizes → Luiza: 4GB Orion1: 8GB Guarani: 5G

Expected behavior: Bigger batches, better performance

Big and Medium Inputs Execution Time

Execution time for big and medium inputs

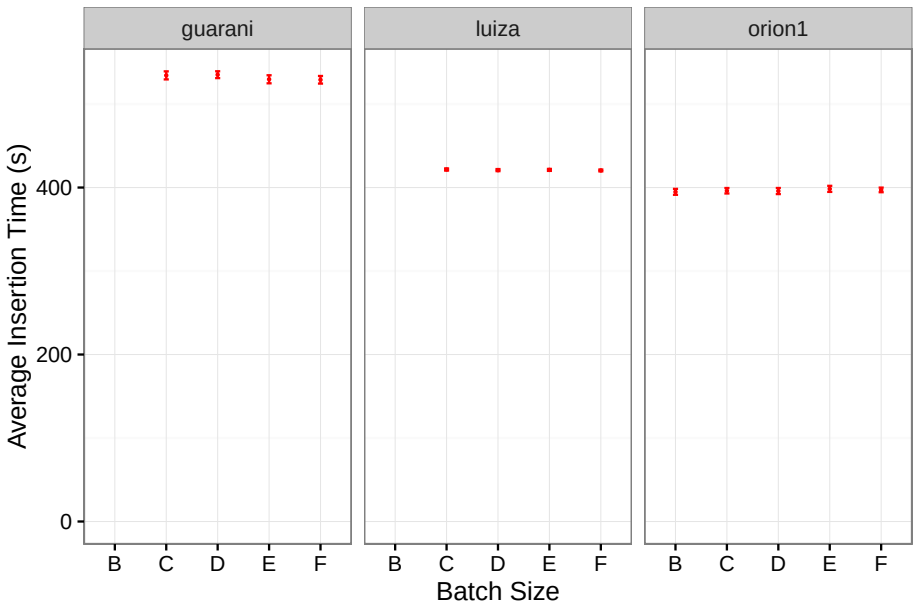


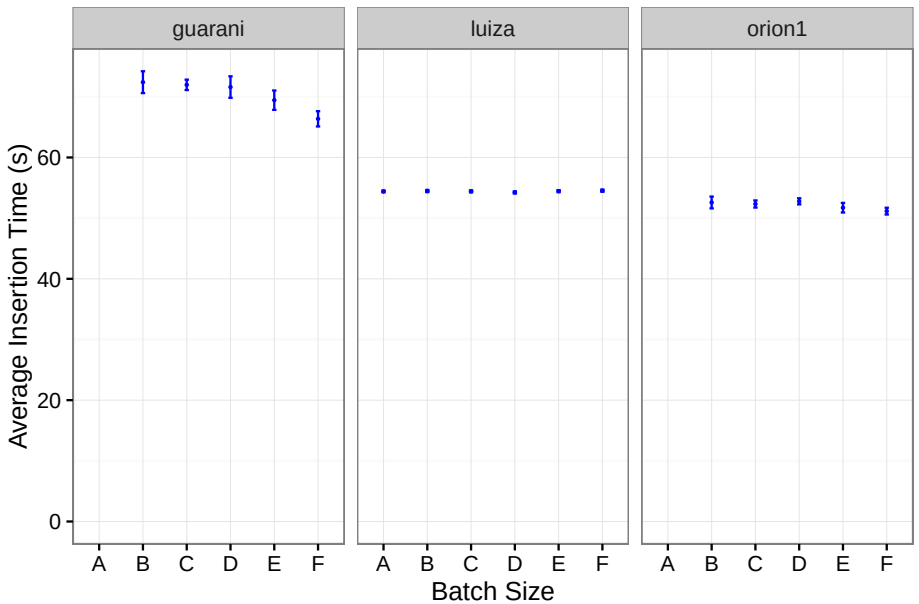


- Batch sizes without results: Java heap space exceeded
- Bigger batches increase execution time - GC overhead

Big and Medium Inputs Insertion Time

Insertion time for big and medium inputs

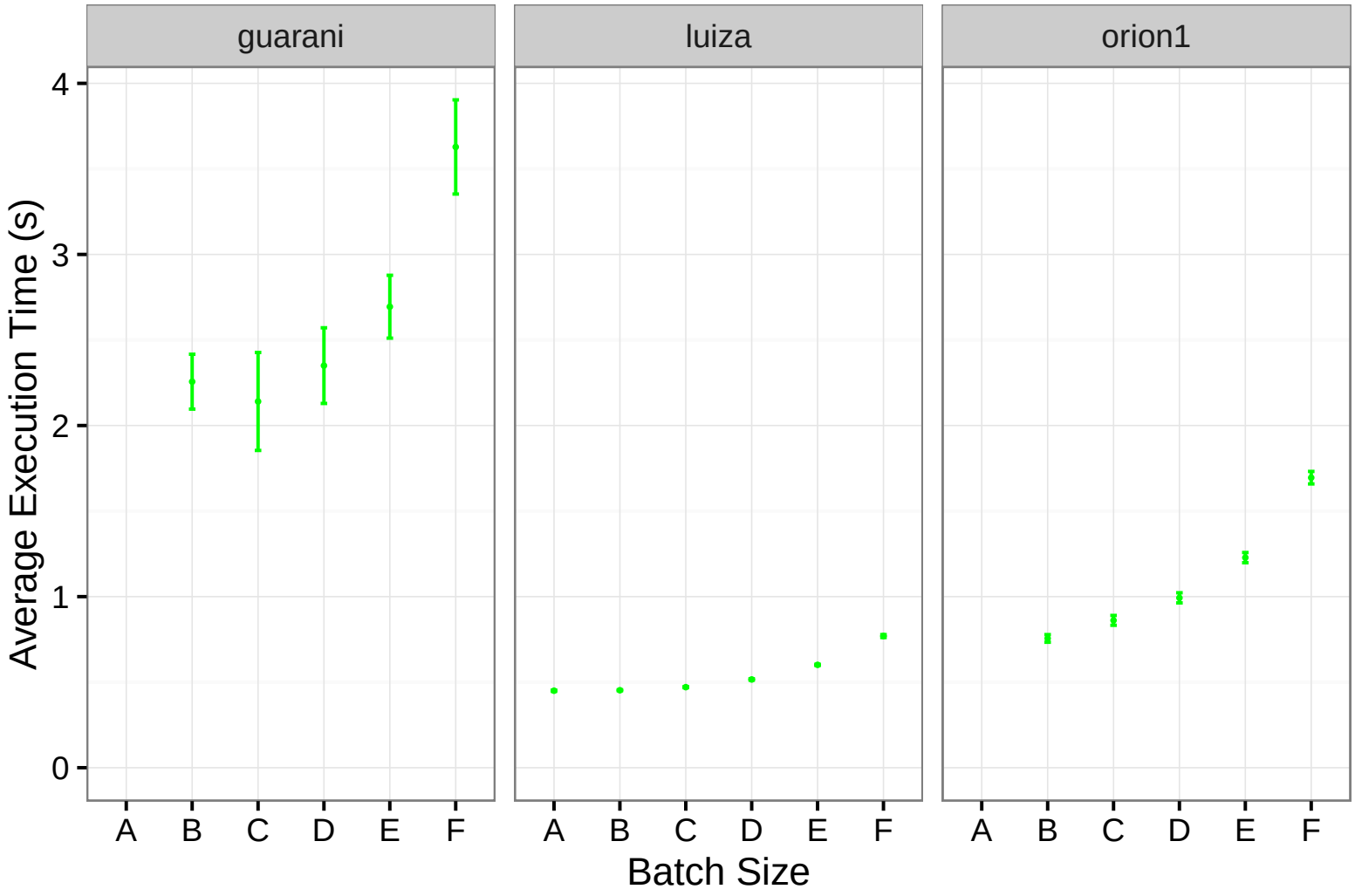




- Constant with different batch sizes

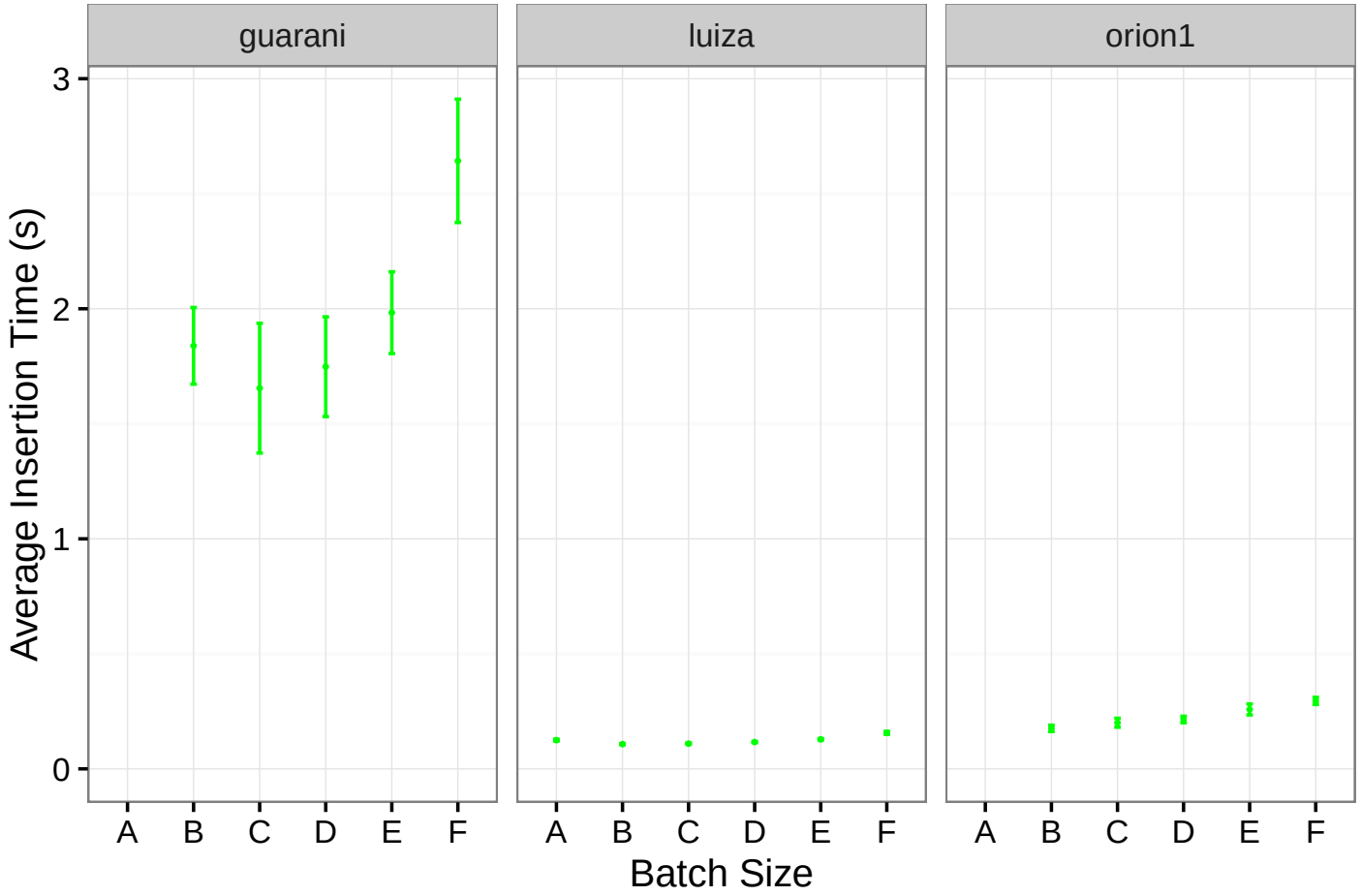
Small Input Execution Time

Execution time for small input



Small Input Insertion Time

Insertion time for small input



- As expected, more insertions increase the insertion time

2.1 Considering a networked environment

Scenario with a remote connection (local network at INF)

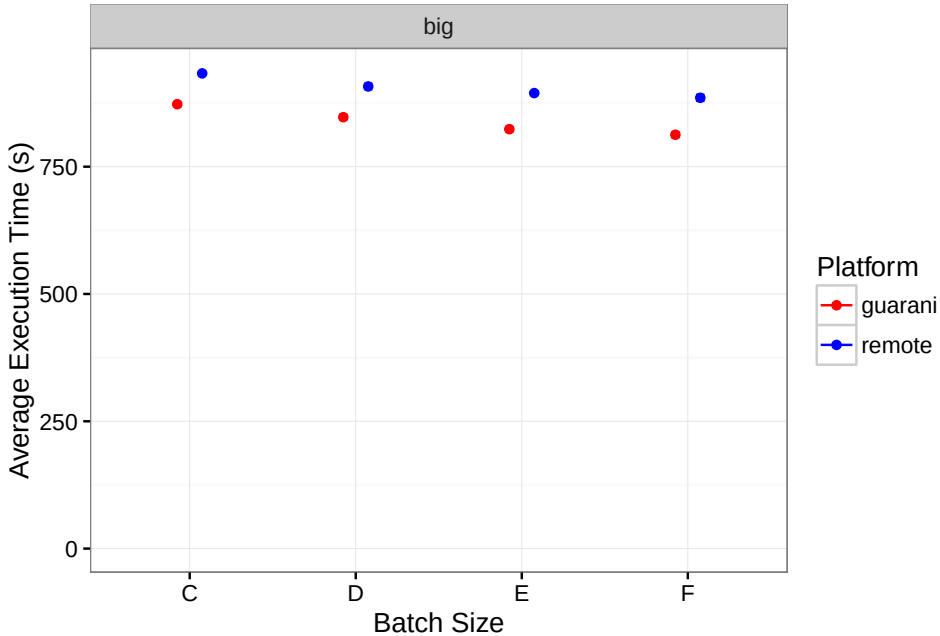
- Ethernet Gigabit, but limited to 100Mbit/s

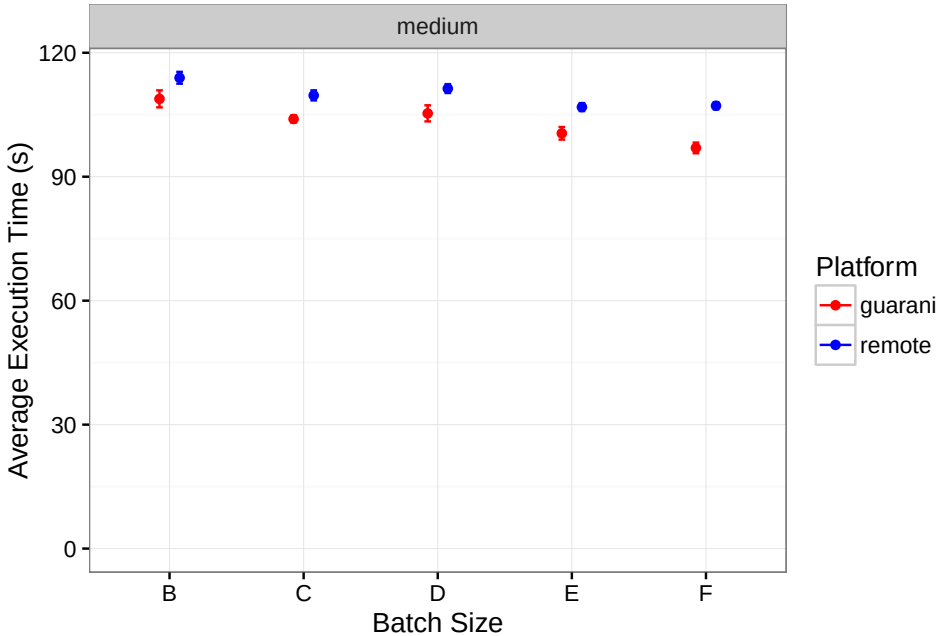
Comparison of different batch sizes

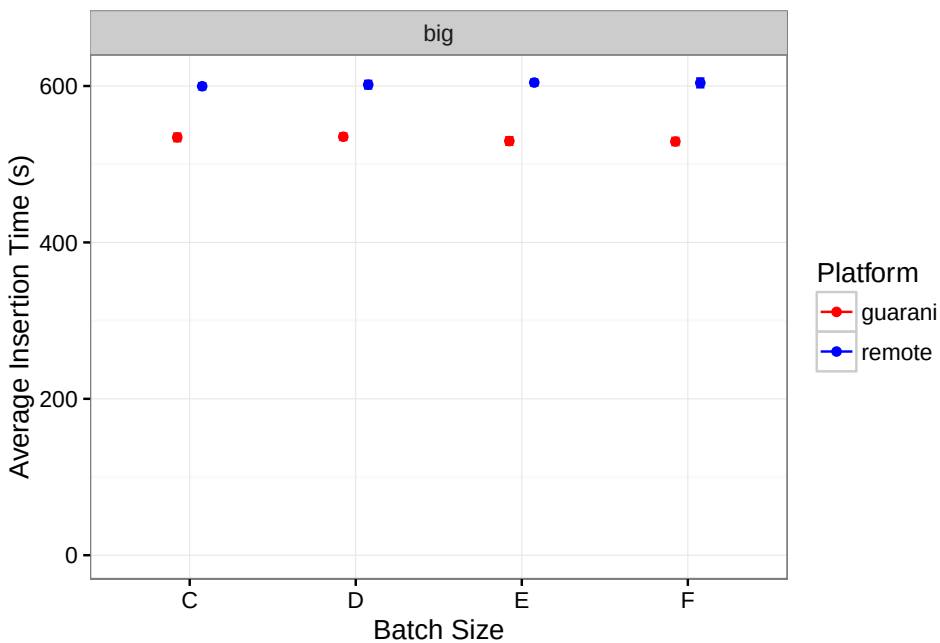
- Local: both Simulator and Database in Guarani
- Remote: Simulator in Guarani, Database server in Orion1

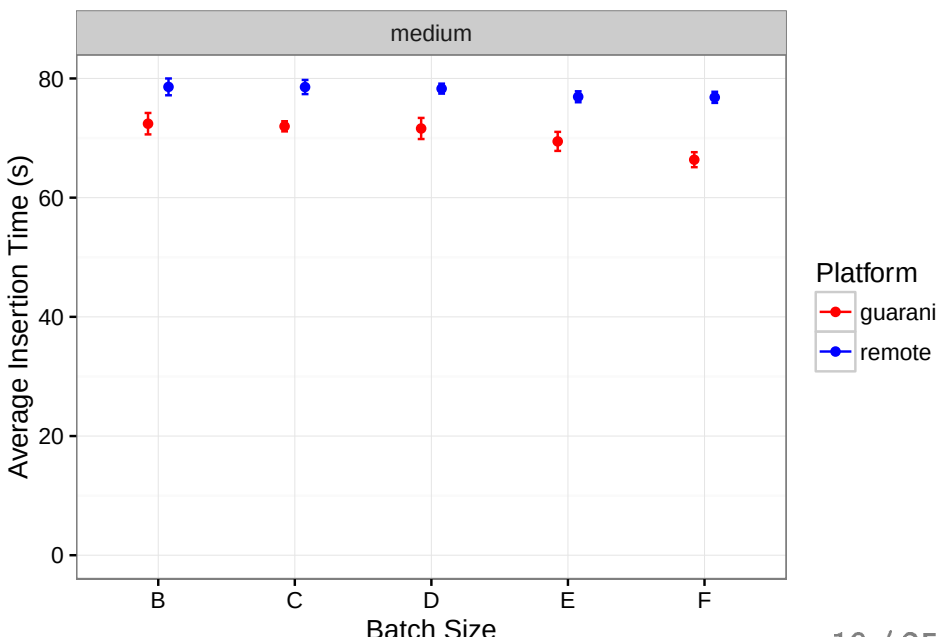
Expected behavior: local execution is always faster (network cost)

Big/Medium Inputs Exec./Insertion Time (remote)



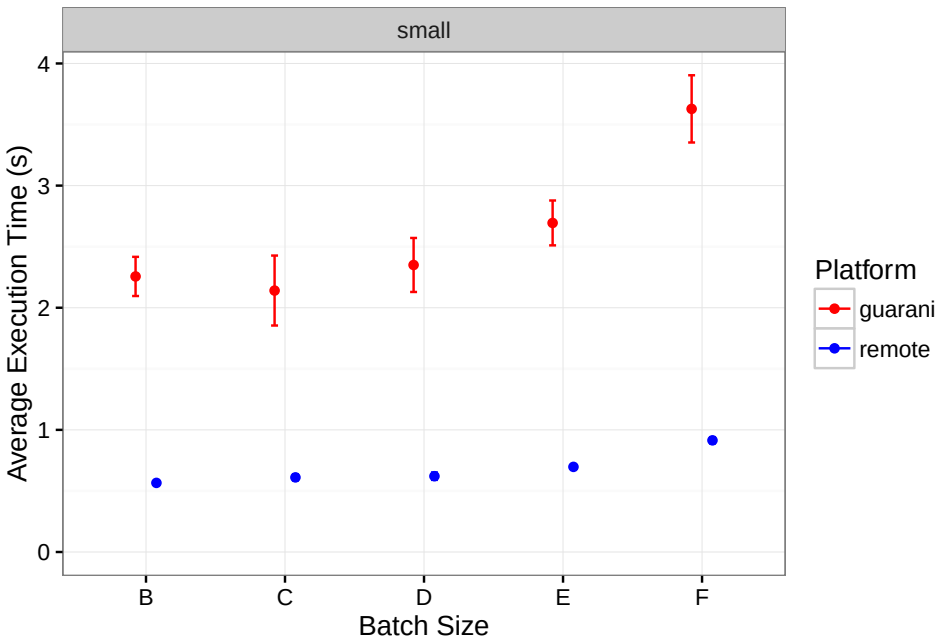


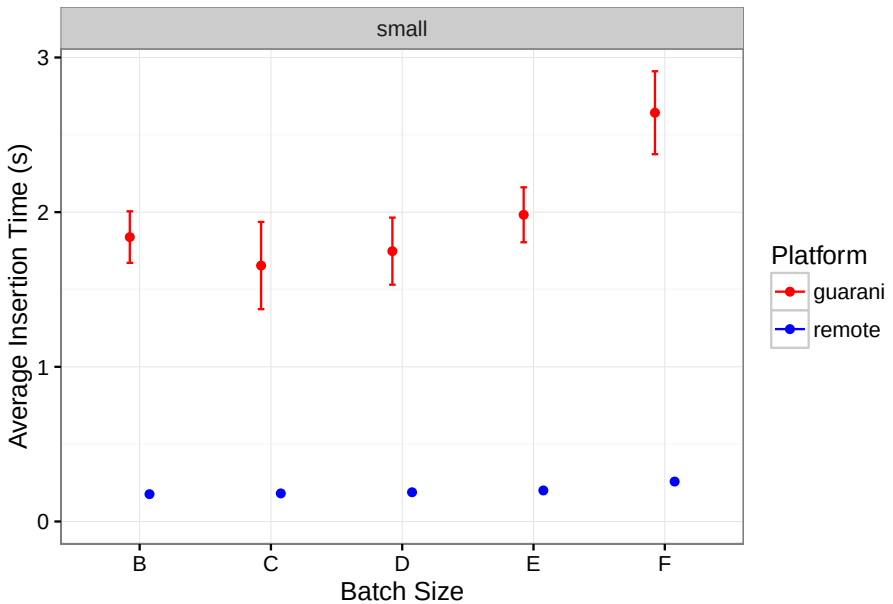




Small Input Execution/Insertion Time (remote)

Execution and Insertion time of remote against local for small input



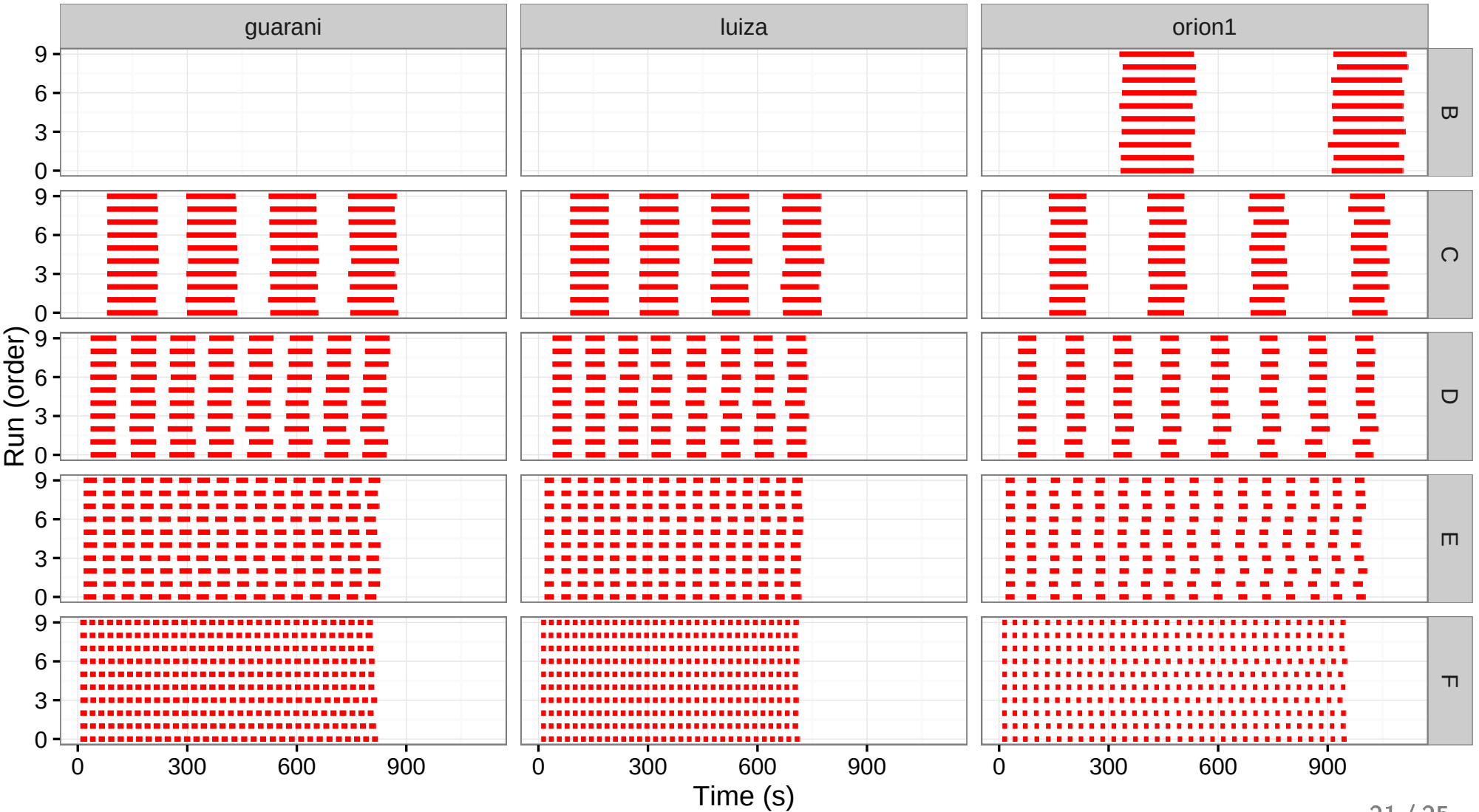


Note: inverse behavior (remote faster than local) for smaller inputs

Batch Insertion Traces - Big Input

Timeline of batch executions for big input

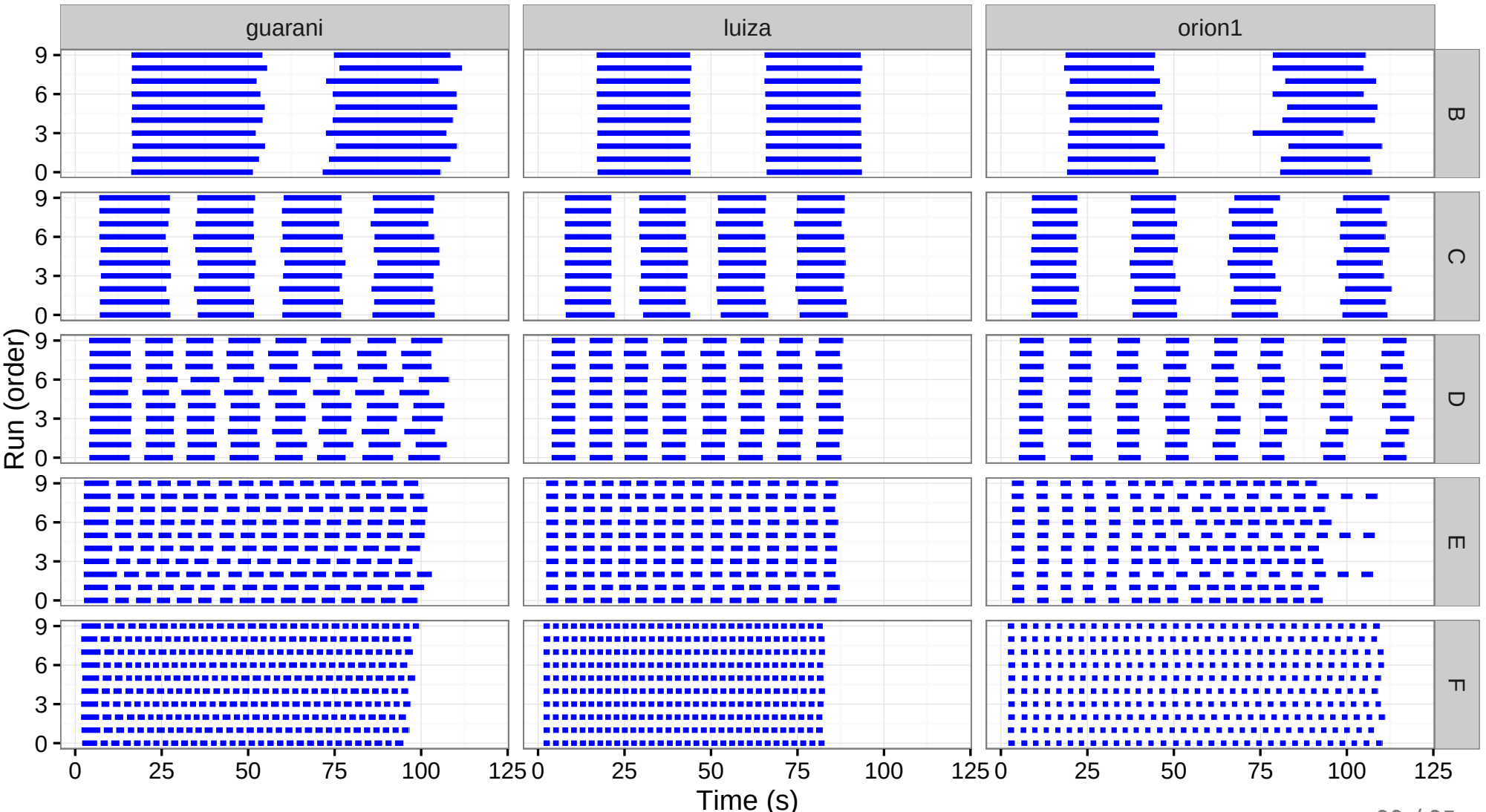
- Note that 10 executions are depicted (in the vertical dimension)



Batch Insertion Traces - Medium Input

Timeline of batch executions for medium input

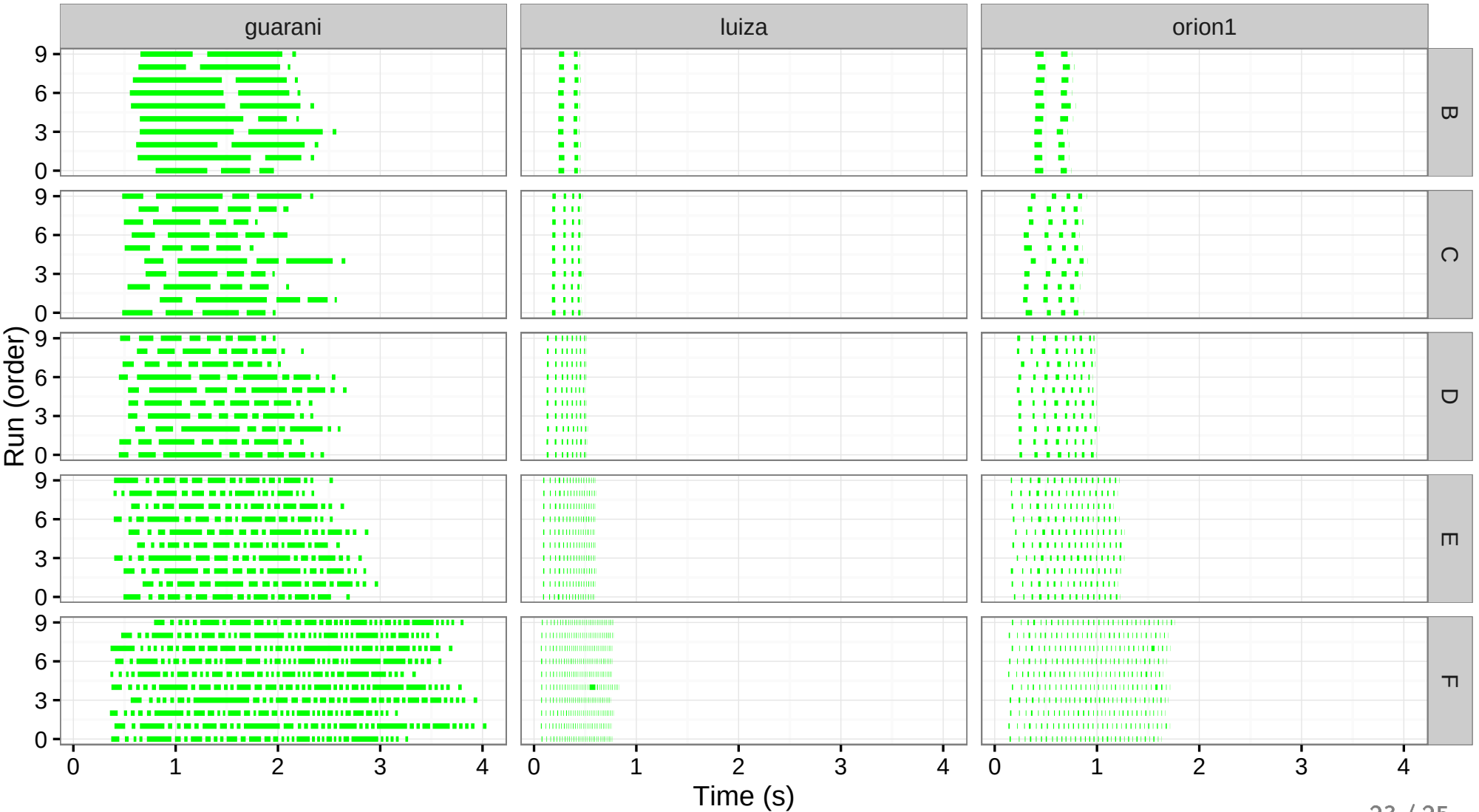
- Note that 10 executions are depicted (in the vertical dimension)



Batch Insertion Traces - Small Input

Timeline of batch executions for small input

- Note that 10 executions are depicted (in the vertical dimension)



Conclusion and Future Work

- Proposal contributions
  - Detach core simulator from entities manipulation
  - Standard plugins implementation
    - Important for proposal validation
  - Resolution of PajeNG issues

Conclusion and Future Work

- Proposal contributions
  - Detach core simulator from entities manipulation
  - Standard plugins implementation
    - Important for proposal validation
  - Resolution of PajeNG issues
- Point to improve
  - Plugin entry points more generic

Conclusion and Future Work

- Proposal contributions
  - Detach core simulator from entities manipulation
  - Standard plugins implementation
    - Important for proposal validation
  - Resolution of PajeNG issues
- Point to improve
  - Plugin entry points more generic
- Future work
  - Support plugins written in other languages
  - Database schema evaluation
  - Further study on batch sizes

Thank you + Propaganda

- Code publicly available at: http://github.com/taisbellini/aiyra
- Documentation of development and experiments in LabBook.org
  - Tutorial on how to compile and use
- Doubts? Contact at tais38@gmail.com