Interactive Historical Data Analysis

Javi Carretero, Technical Architect
TrendMiner
04/06/19

Plan

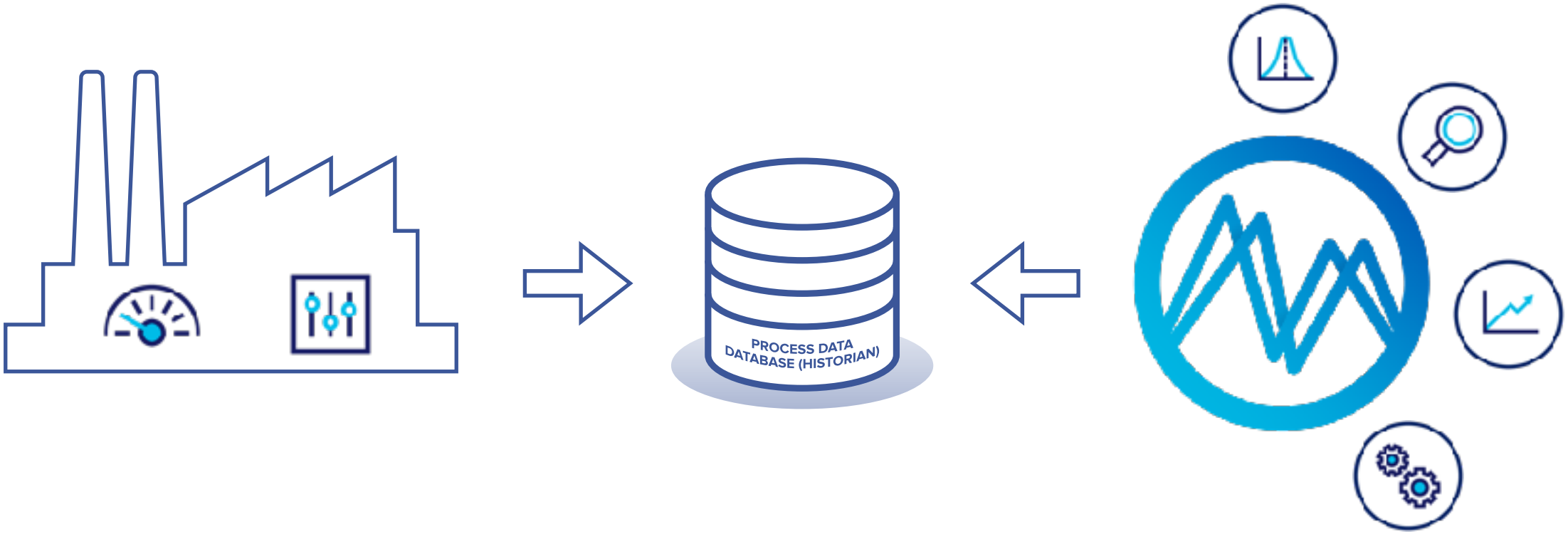
- Introduce TrendMiner
- Discuss context & user needs
- TrendMiner 1.0
- TrendMiner 2.0 (with *Apache Ignite*)
- Challenges
- Future work

TrendMiner

Empower process and asset **experts** with **advanced analytics** to Analyse, Monitor and Predict the operational performance of batch, grade and continuous manufacturing processes.

We **democratise analytics** by giving insights to the people who need answers: the **engineers** and **operators** in the plant.

TrendMiner



TrendMiner

Descriptive Analytics

Predictive Analytics

Modelling

TrendMiner

Context & Scale

- > 300M points per time series
- 10-40K active time series
- Source of data is generally very slow!

User Expectations

- Time to first result < 1s
- Higher resolution
- More active time series
- More advanced analytics

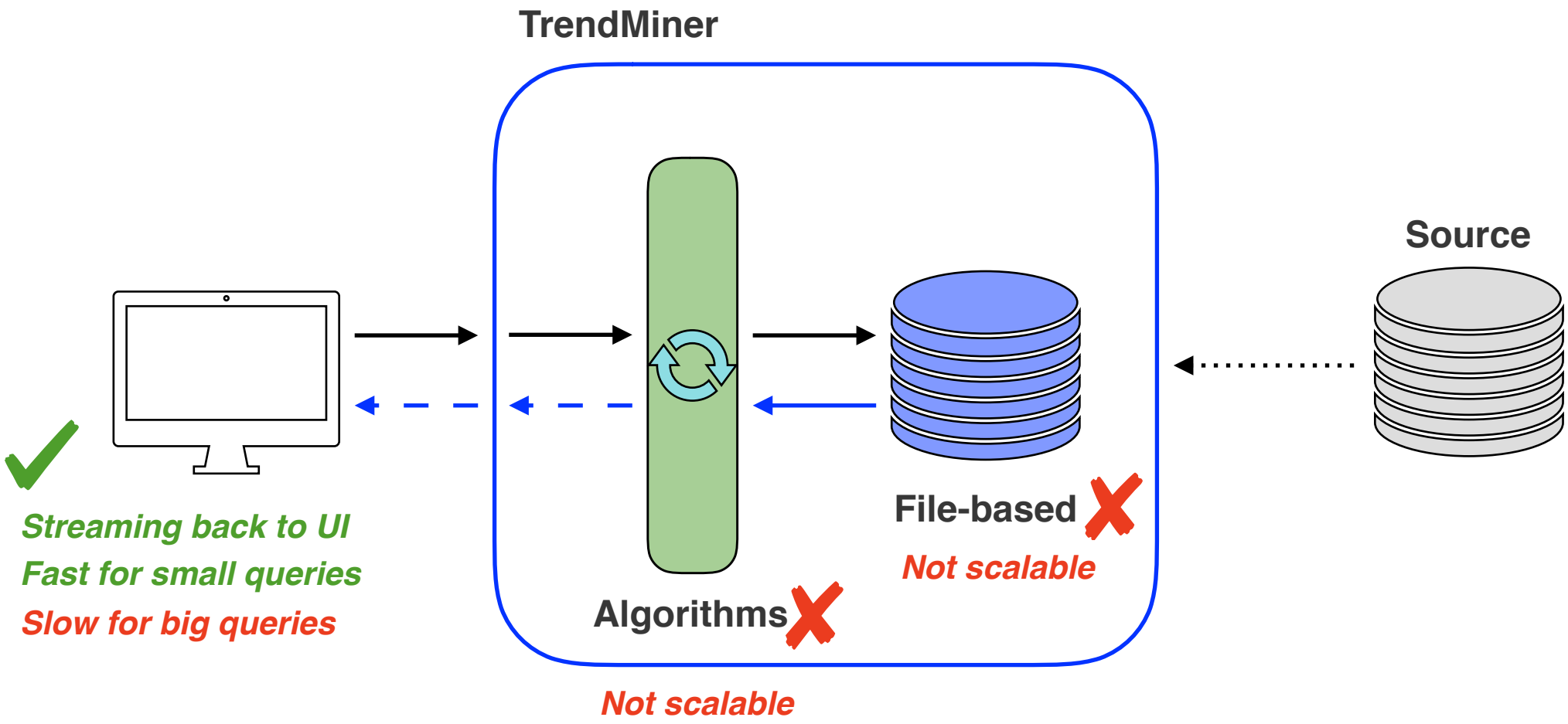
Responsiveness

Overall performance

TrendMiner 1.0

Focus on responsiveness (making TrendMiner more interactive)

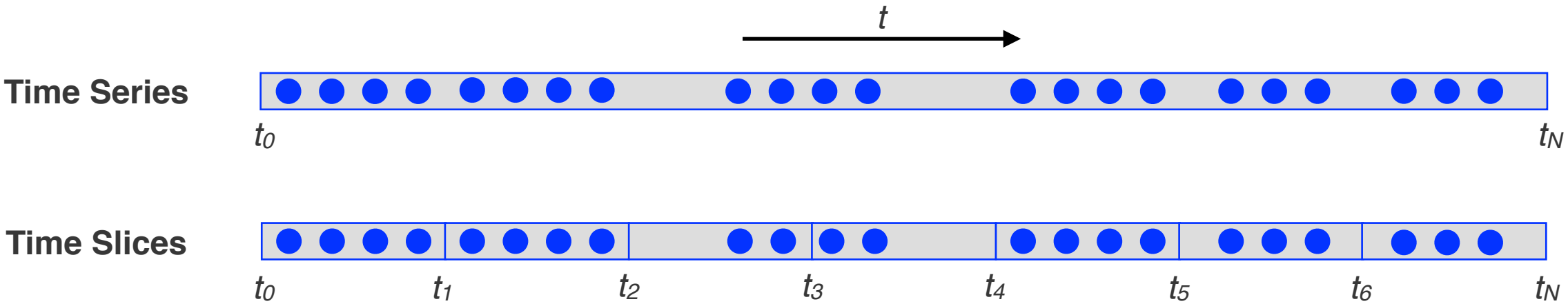
TrendMiner 1.0



TrendMiner 2.0

Focus on performance (making TrendMiner more efficient)

TrendMiner 2.0 - Caching



IgniteCache<Key, Data>

startDate (t_i)
endDate (t_j)
timeSeriesId

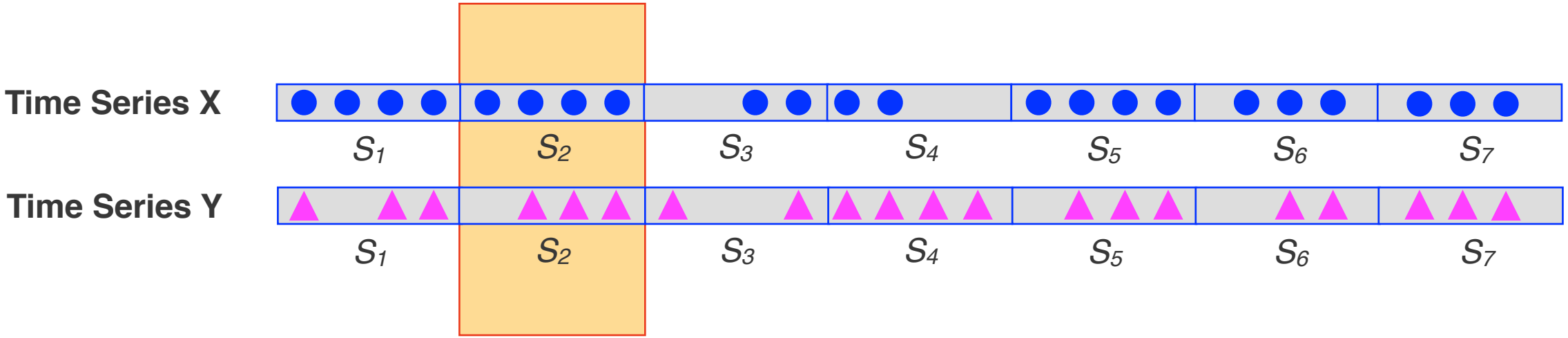
→

[ts_0, $value_0$]
[ts_1, $value_1$]
[ts_2, $value_2$]

TrendMiner 2.0 - Caching

2019-06-04	**16:20:15.165**	*Point timestamp*
2019-06-04	**16**:20:15.165	*Slice by hour*
2019-06-04	16:20:15.165	*Slice by day*
2019-06-04	16:20:15.165	*Slice by month*
2019-06-04	16:20:15.165	*Slice by year*

TrendMiner 2.0 - Affinity

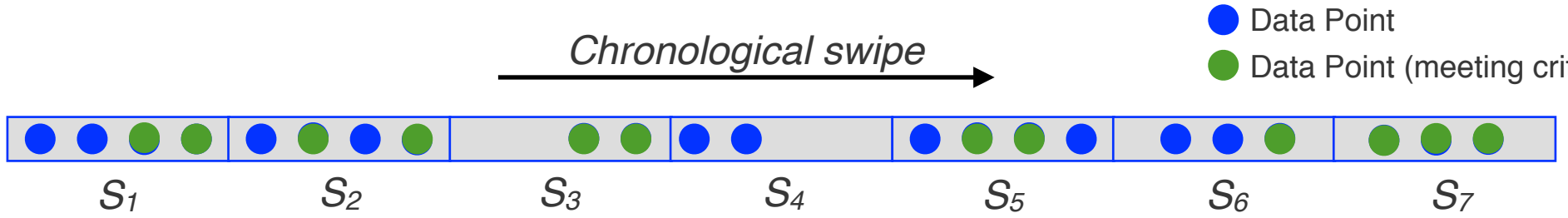


S_i ⟶ $Node_j$

(all Time Series)

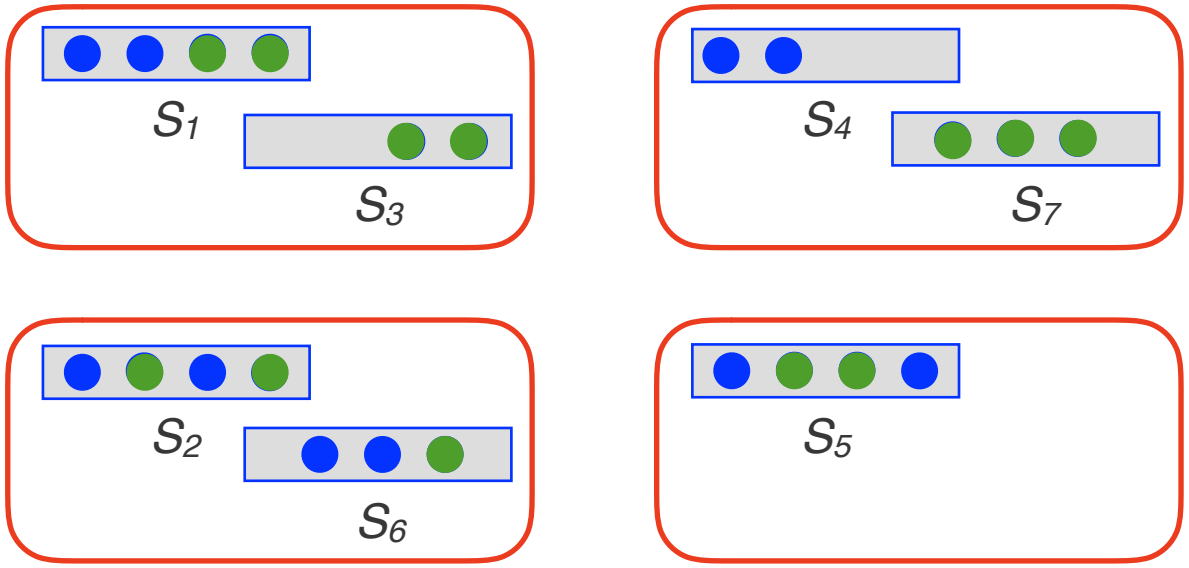
e.g: 2019-06-04

TrendMiner 2.0 - Compute Grid

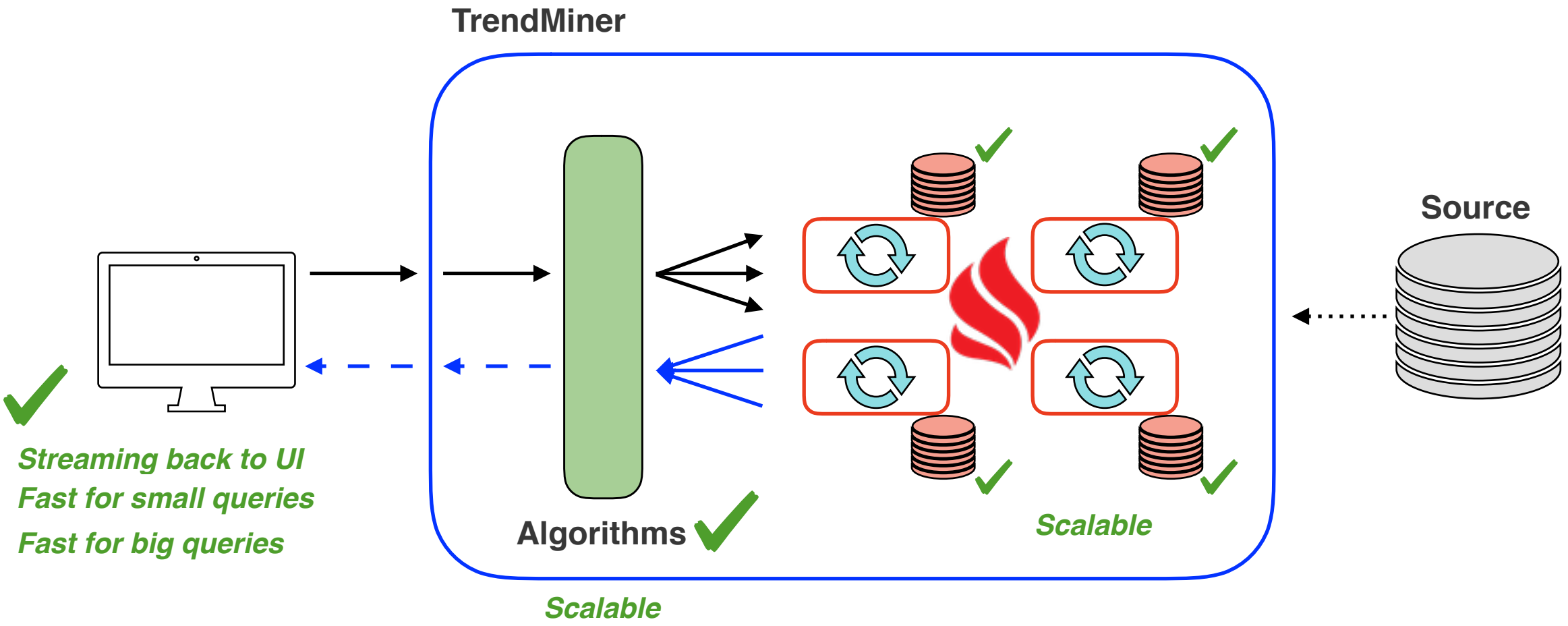


Scalable algorithms = *IgniteCompute*

A. Split search (affects N slices)
B. Scatter "jobs" = *affinityCall*
C. Chronological swipe (single slice)
D. Post-process partial results (e.g. merging)



TrendMiner 2.0 - Result



Challenge - Multi-level Prioritisation

Search dimensions

❖ Time Series (single **vs** multiple series)

CPU usage

❖ Search window (single **vs** multiple time slices)

Ignite jobs

❖ Algorithm (visualisation **vs** descriptive **vs** predictive analytics)

Urgency

Challenge - Multi-level Prioritisation

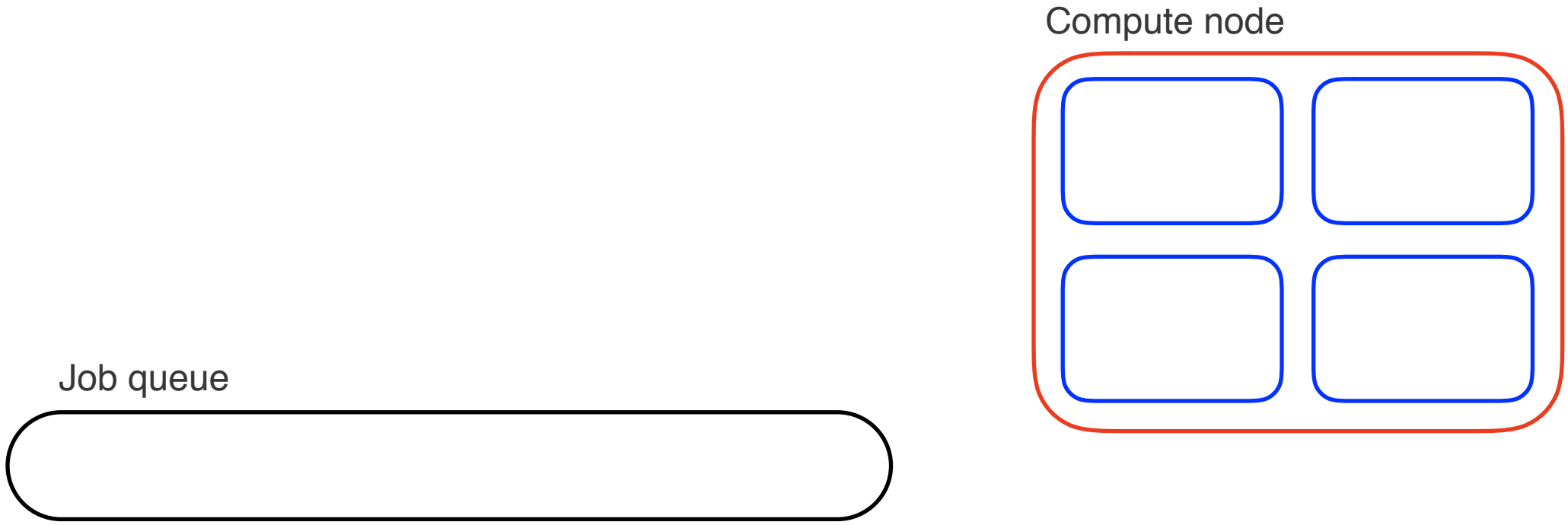
Ignite Capabilities

- *PriorityQueueCollisionSpi* (*grid.task.priority*) = **1 dimension!**

MultiLevelPriorityQueueCollisionSpi (custom implementation)

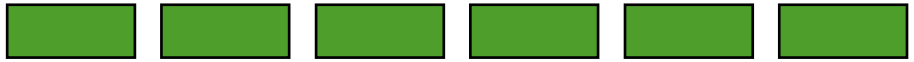
- Still use *grid.task.priority*
  - Priority degression factor = #(Time Series) X #(Time Slices)
- Urgency via "Service Levels" (0...N) = **2nd dimension**

Multi-level Prioritisation (Example)

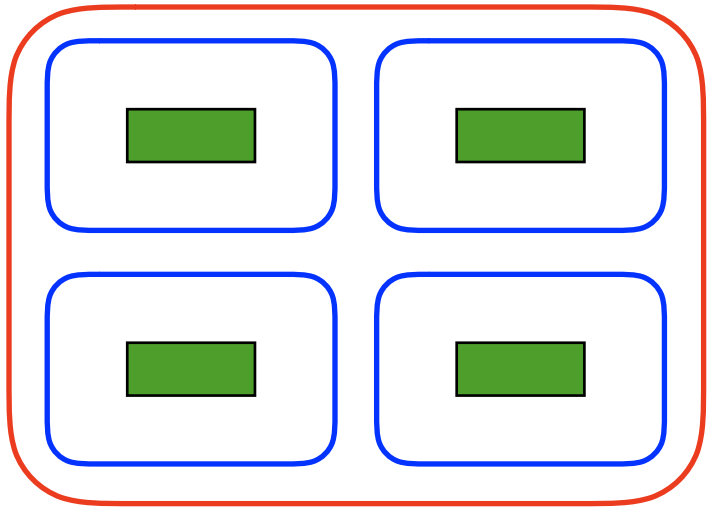


Multi-level Prioritisation (Example)

Historical search = Service Level 1



Queued tasks have a priority (degression factor)



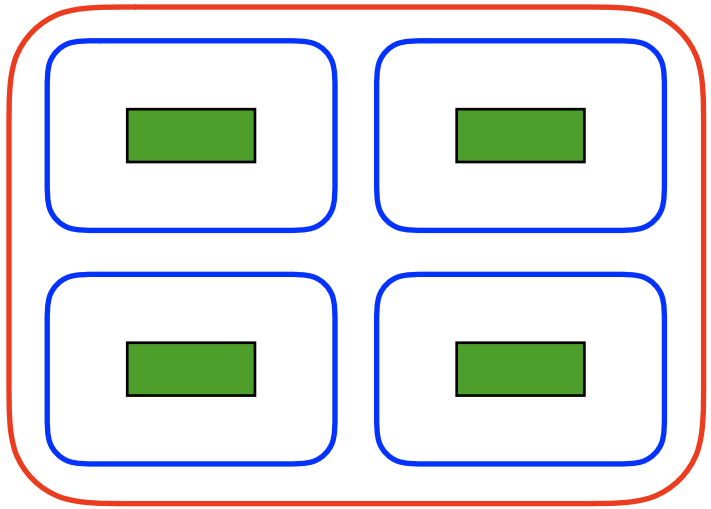
Compute node

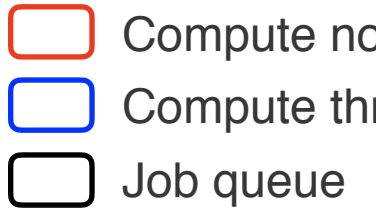
Compute thread

Job queue

Multi-level Prioritisation (Example)

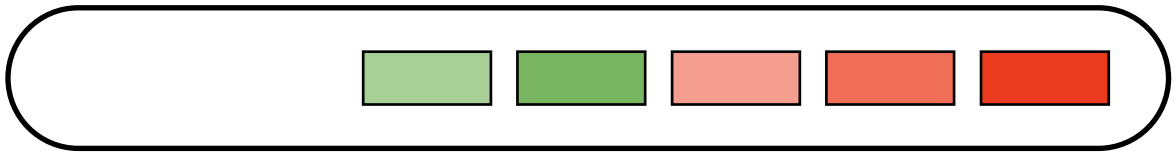
New computation (max urgency) = Service Level 0

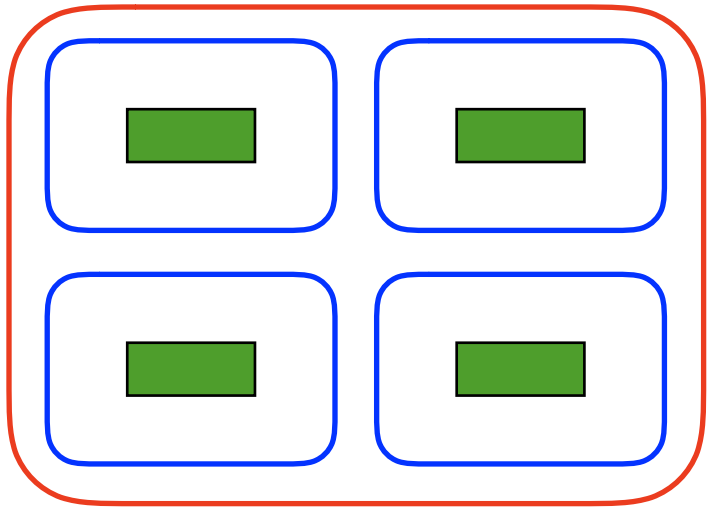


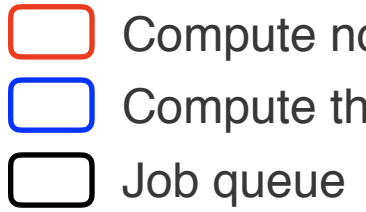


Multi-level Prioritisation (Example)

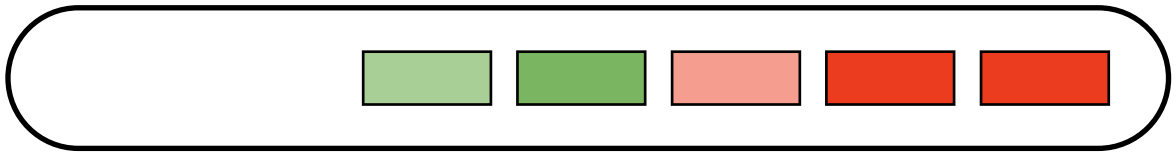
New computation (max urgency) = Service Level 0

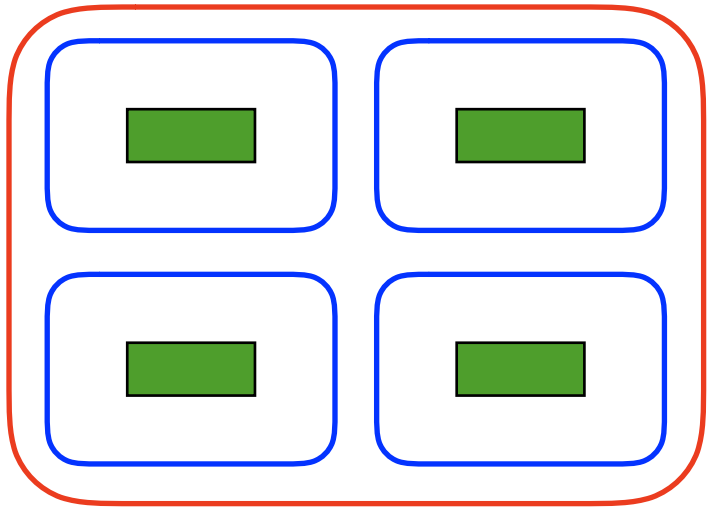


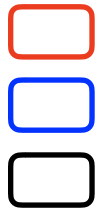




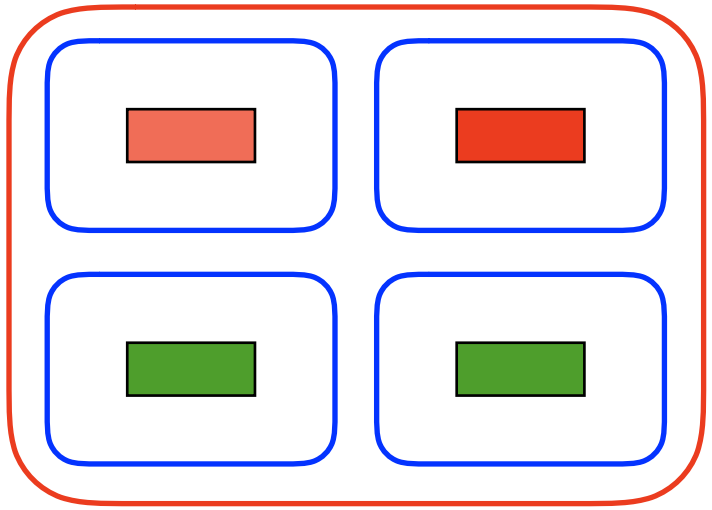
Multi-level Prioritisation (Example)

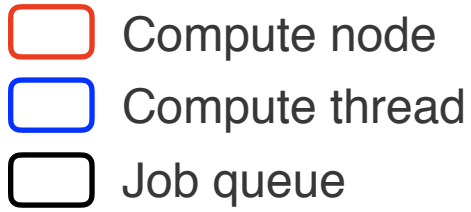






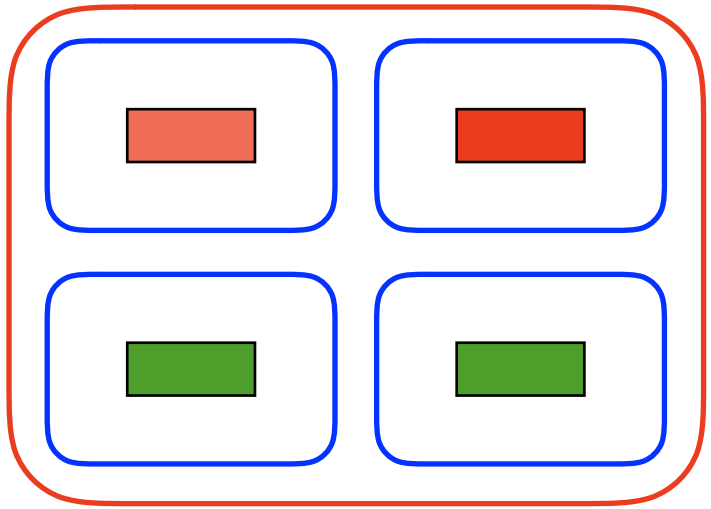
Multi-level Prioritisation (Example)

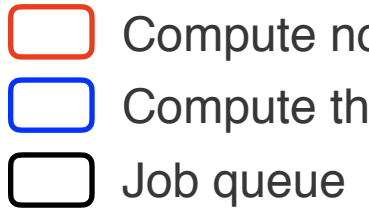




Multi-level Prioritisation (Example)

Historical search = Service Level 1

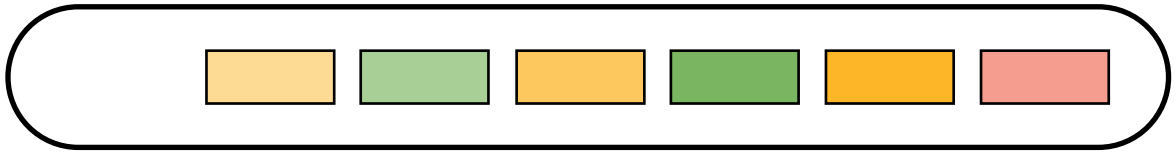


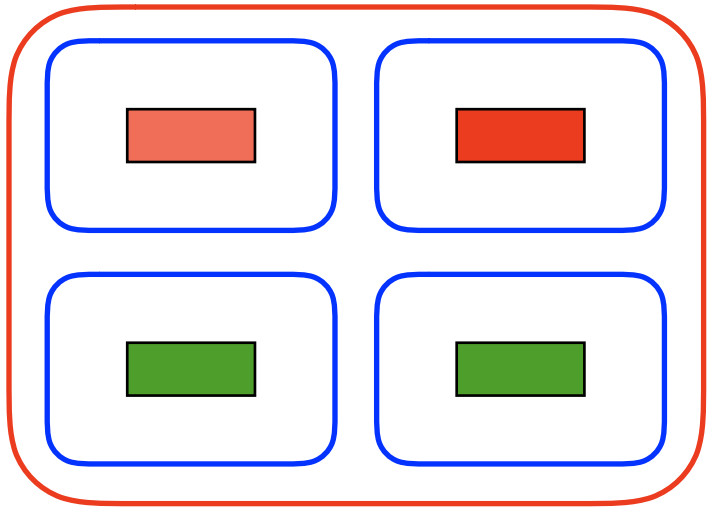


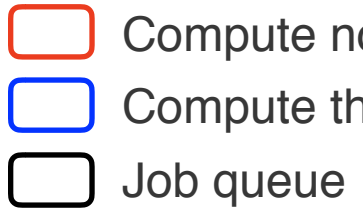
Multi-level Prioritisation (Example)

Historical search = Service Level 1

First task > priority (no degression factor applied)







Future Work

- Improve scheduling efficiency (predictable job runtime)
- Prevent job starvation (e.g. job-stealing SPIs)
- Make all algorithms scalable
- Pave way for Ignite Native Persistence

Thank you!

javi.carretero@trendminer.com