DISTRIBUTED COMPUTATION OF CRITICAL PATH

Pierre-Frédérick DENYS

Monday 16 May 2022

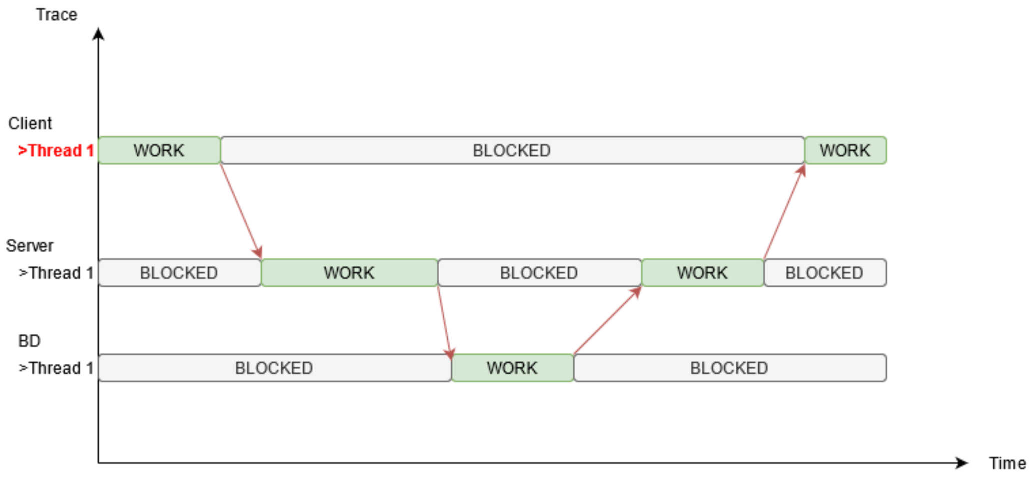
Agenda

- Introduction
- Challenges about critical path computation
- Proposed algorithm
- Future work and usecases
- Conclusion

“

Quick reminder about Critical Path

Reminder about critical path



- The data structure as a two-dimensional doubly linked list, where horizontal edges are labelled with task states, and where vertical edges are signals between tasks (either a wake-up or a network packet)

- The active path of execution is the execution path where all blocking edges are substituted by their corresponding subtask

Critical path usage

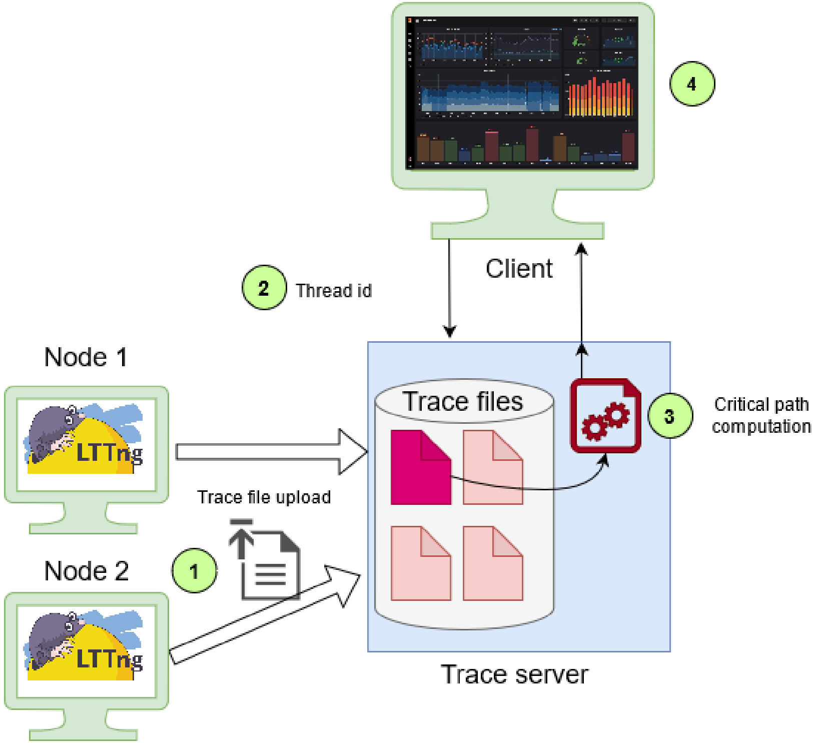
- Need for large distributed systems tracing
  - HPC systems
  - MPI clusters
  - Kubernetes and container clusters
- Critical path computation not optimized
- Transfer of trace files on analysis node is mandatory
- Critical path unavailable in Theia and Grafana plugin

“

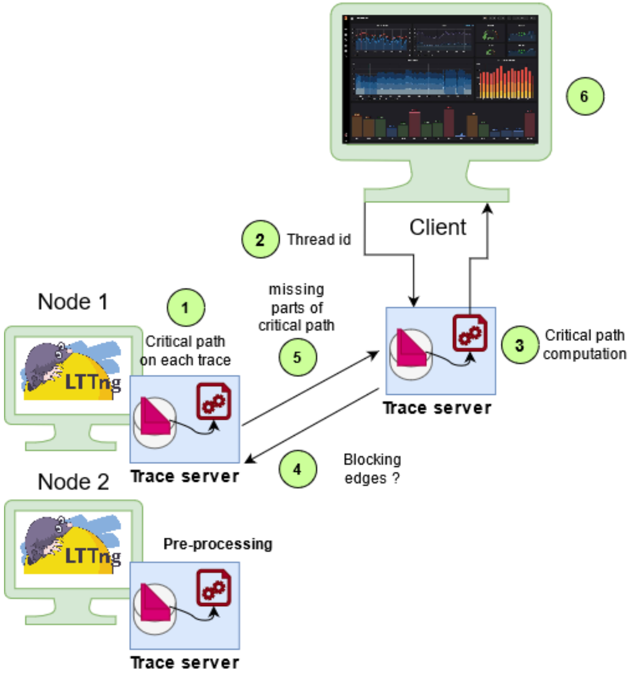
Actual architecture and challenges

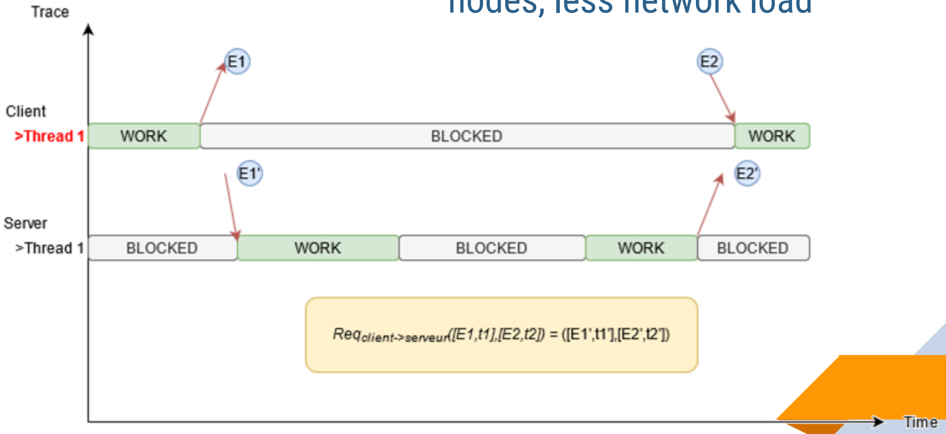
Actual architecture in Trace Compass

-



Parallelisation of the computation : algorithm





- Pre-processing of critical path on each node

- On client request, process the critical path of the trace, and ask only the missing parts of the path to other nodes

- Distributed processing, suitable for large number of nodes, less network load

My work

- Improve algorithm to :

Compute execution graph, outbound and inbound edges of critical path of each trace independently on computing nodes

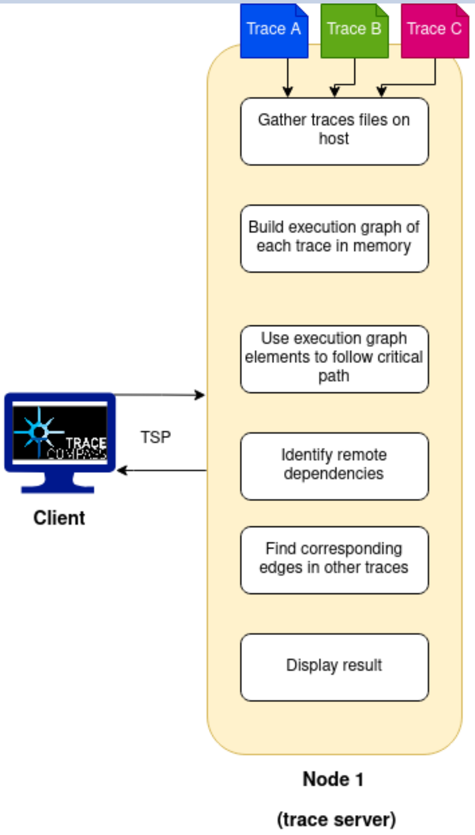
Be able to process the full critical path from pre-processed parts on viewer node
Send minimum data between computing and viewer nodes

- Storage of critical path on disk :

Usage of state system for horizontal edges and segment store for vertical ones
Storage on disk rather than in memory*

(*related work done by Arnaud and Geneviève)

My work



Trace A

Trace B

Trace C

Gather traces files on host

Build execution graph of each trace in memory

Use execution graph elements to follow critical path

Identify remote dependencies

Find corresponding edges in other traces

Display result

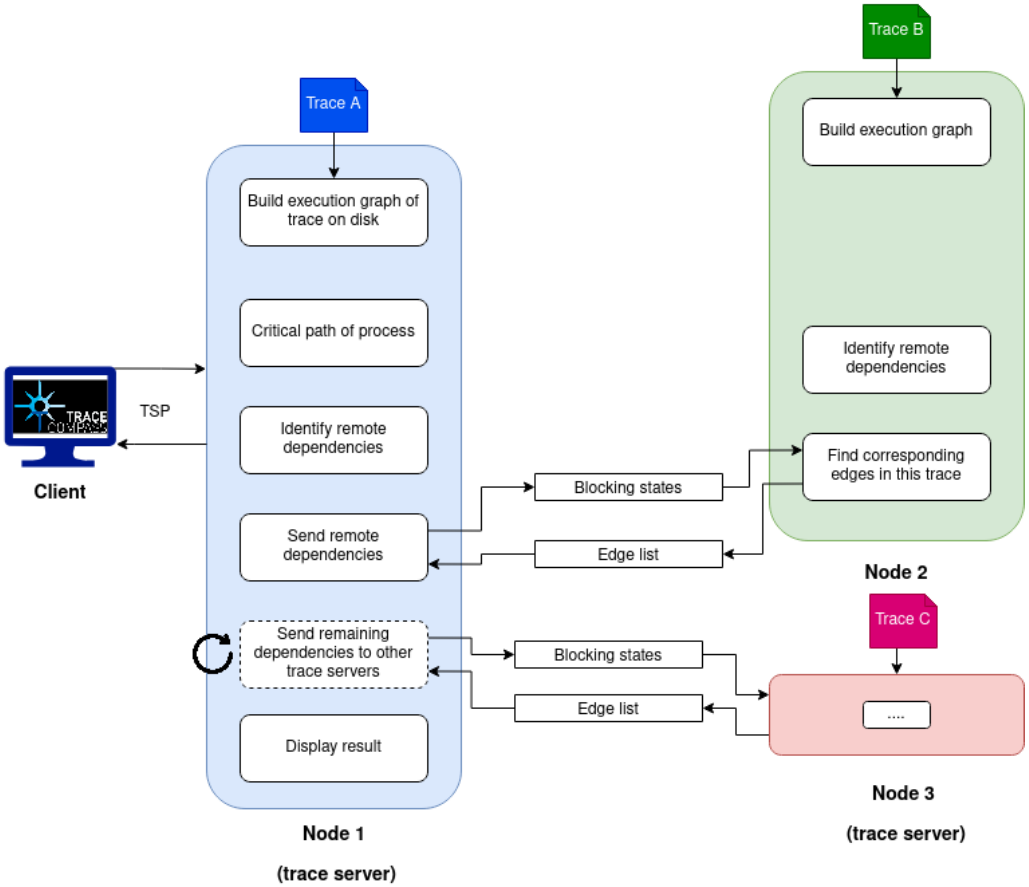
TSP

Client

Node 1

(trace server)

Actual algorithm



Trace A

Build execution graph of trace on disk

Critical path of process

Identify remote dependencies

Send remote dependencies

Send remaining dependencies to other trace servers

Display result

TSP

Client

Node 1

(trace server)

Blocking states

Edge list

Blocking states

Edge list

Trace B

Build execution graph

Identify remote dependencies

Find corresponding edges in this trace

Node 2

Trace C

....

Node 3

(trace server)

Distributed version

Demo

Demo

New algorithm :
open scp_src trace on host
and
scp_dest on distant machine

https://youtu.be/ofc34w3uv5M

“

Future work and usecases

What remains to be done ?

- Test and characterize overhead and efficiency of new distributed method on several usecases
- Integration of communication between nodes in Trace server Protocol

Usecases

- Target usecases :
  - MPI cluster : follow a MPI task between computing nodes
  - Kubernetes cluster : follow a request in a distributed web application
  - ZeroMQ communication : follow a message exchange between several containers

“

Conclusion

Conclusion

- Parallelisation of critical path computation
- Next step : Integration of Critical path in Trace Server Protocol (for Theia and Grafana viewers)
- Extend parallelisation to other kind of analysis

Thank you for listening !