Towards high-level programming for distributed systems

Laurent Prosperi

October 29, 2019

Programming distributed system

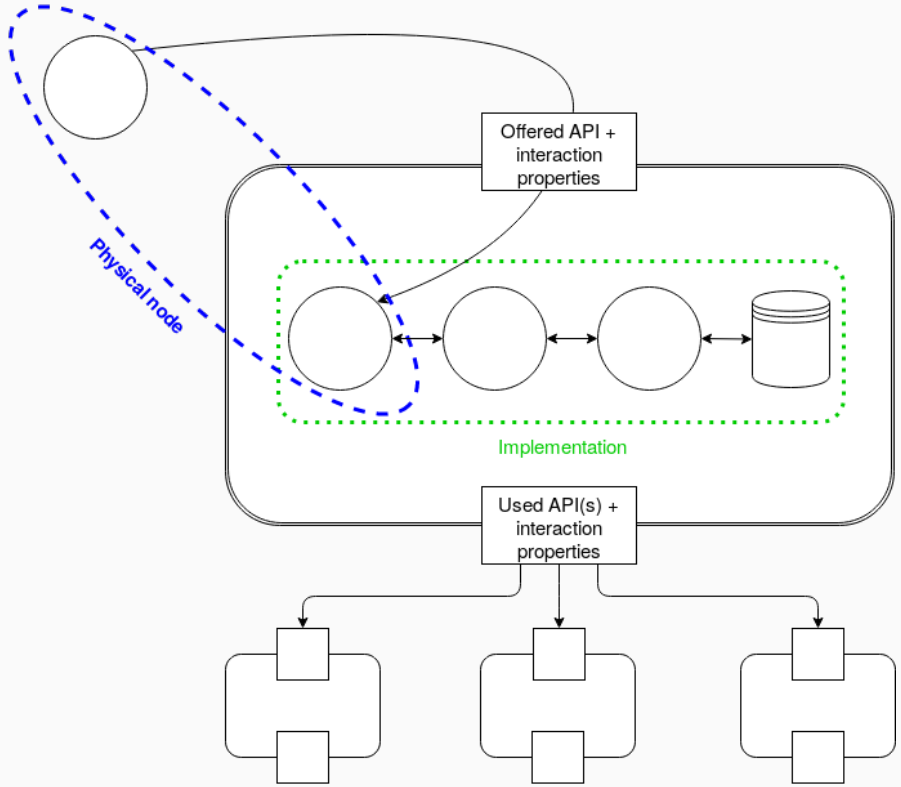
Programming distributed system is hard

- lots of issues
- lots of trade-offs

Requirements depend of the audience

- hide complexity -> productivity
  - transparent fault-tolerance
  - transparent consistency
  - transparent elasticity
- vs building systems -> control
  - building fault-tolerance
  - building consistency
  - building elasticity

Our approach



Fine-grain control

Power to create distributed abstractions

Compose

Encapsulate

Why composition is so important ?

Separation of concerns, modularity

Ex: communication layer, metadata layer, isolation layer, security layer

Glue language (external DB to store data object)

Modular, composable verification techniques

- at the boundary -> verify at boundaries (proof, dynamic checking, test)
- "hierarchy" of verification steps

Encapsulation

Abstractions' interactions (message/event, shared variable, RPC)
Encapsulation

- Restrain access between abstractions
- An abstractions should not disrupted by other un-related abstractions
- Restrain access between abstractions (restrict communication channels)

Describes by the interface of the abstraction

Runtime behaviours as first class abstraction

Runtime = build as a set of abstractions (+ bootstrap code)
Take adventage of the properties of the language
Programmers can control the runtime with first class abstractions

Runtime layers

1. Operators + communication links between them
2. Metadata piggy-packing (provenance, ...)
3. Isolation management
4. Error-handling, fault-tolerance
5. Placement management (annotation)
6. Elasticity

A language for high level distributed computing

1. Target system developers first
    - Expose runtime primitives as first class language abstractions
    - Create a hierarchy of abstractions, extended by programmers
2. Increase robustness of distributed programs
    - Correctness of compositions
    - Encapsulation
3. Make it practical
    - Reuse existing code: DSL
    - Reuse existing tools/systems (e.g. DB) : Glue language