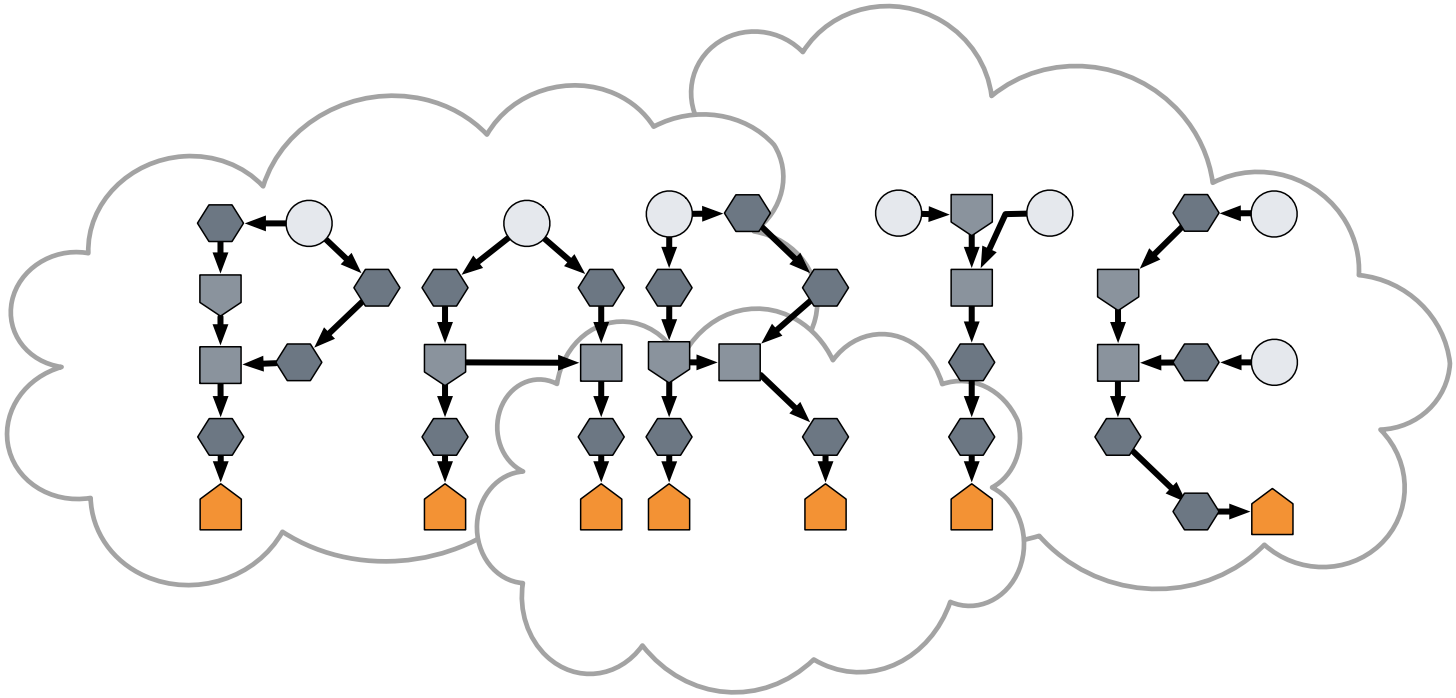


Cloud PARTE

Elastic Complex Event Processing based on Mobile Actors

J. Swalens, T. Renaux, L. Hoste, **S. Marr**, W. De Meuter

Software Languages Lab

Vrije Universiteit Brussel

AGERE Workshop, 2013-10-27

Use Case

Traffic Management

Traffic Management at City Scale?

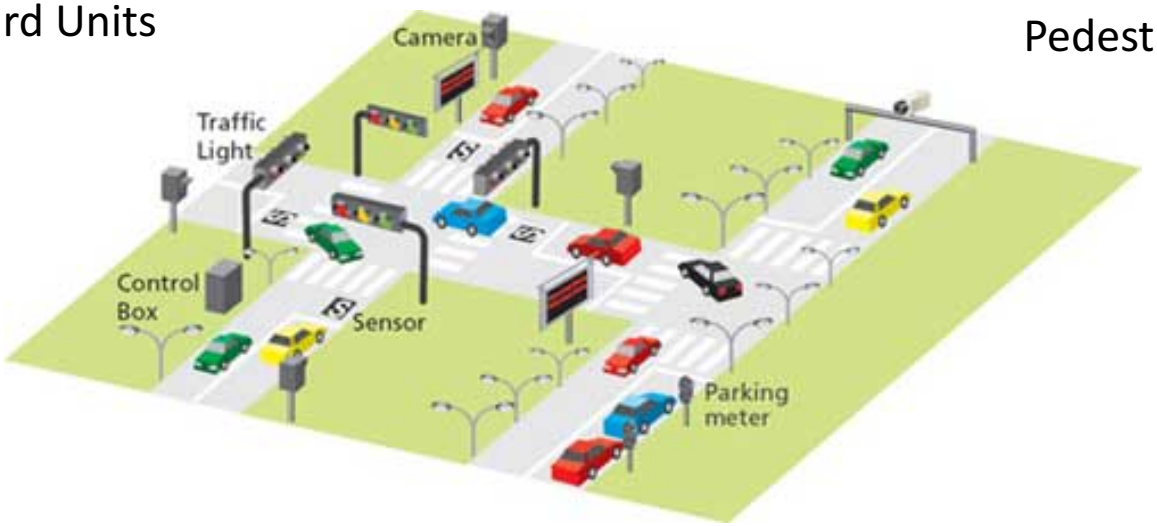
Singapore: GPS coord. of 16000 taxis = 200 coord./sec = 12KB/sec

Source: City-Scale Traffic Estimation from a Roving Sensor Network, Javed Aslam, Sejoon Lim, Xinghao Pan, and Daniela Rus. Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems. ACM, (2012)

Challenges:
Large Varity of Input Sources

Personal Onboard Units

Pedestrian Sensing



Vehicle Sensing

Challenges: Combining the Data

Imperative

```
int gru_tapping(struct grail *ge, const struct utouch_frame *frame) {
    struct gesture_recognizer *gru = ge->gru;
    struct tapping_model *state = &gru->tapping;
    struct move_model *move = &gru->move;
    state->tap = 0;
    if (frame->num_active && !frame->prev->num_active) {
        state->mintouch = 0;
        state->maxtouch = 0;
    }
    if (move->ntouch > state->maxtouch) {
        if (state->active) {
            gin_gid_discard(ge, state->gid);
            state->active = 0;
        }
        state->start = move->time;
        state->maxtouch = move->ntouch;
        set_props(ge->gin, state, move, frame);
        if (state->maxtouch <= 5) {
            int type = GRAIL_TYPE_TAP1 + state->maxtouch - 1;
            state->gid = gin_gid_begin(ge, type, PRIO_TAP, frame);
            state->active = 1;
        }
        return 0;
    }
    if (!state->active) {
        state->mintouch = move->ntouch;
        state->maxtouch = move->ntouch;
        return 0;
    }
    if (move->ntouch <= state->mintouch) {
        int x = state->prop[GRAIL_PROP_TAP_X];
        int y = state->prop[GRAIL_PROP_TAP_Y];
        int t = move->time - state->start;
        if (t > move->fm[FM_X].bar_ms) {
            gin_gid_discard(ge, state->gid);
            state->mintouch = move->ntouch;
            state->maxtouch = move->ntouch;
            state->active = 0;
            return 0;
        }
        state->tap = state->maxtouch;
        state->prop[GRAIL_PROP_TAP_DT] = t;
        gin_gid_event(ge, state->gid, x, y, state->maxtouch,
                  state->prop, state->nprop, 1);
        state->mintouch = move->ntouch;
```

a simple tap

Declarative

```
(defrule detectTap
 (Press (x ?x)
        (y ?y)
        (finger "index")
        (timestamp ?t1))
 (Release (x ?x)
          (y ?y)
          (finger "index")
          (timestamp ?t2))
 (test (< ?t1 ?t2))
 =>
 (tapDetected ?x ?y)))
```

Challenges: Combining the Data

```
(defrule StolenCarInTraffic
  (StolenCar    (plate ?pl))
  (CarInTraffic (plate ?pl) (camera ?c))
  (Camera (id ?c) (highway ?hw)
          (direction ?d) (position ?p))
=>
  (printout "Sloten car " ?pl
            " seen on highway " ?hw
            " at " ?p
            " in direction " ?d))
```

```
(defrule CarTooFast
  (CarInTraffic (camera ?c1)
    (plate ?pl) (time ?t1))
  (CarInTraffic (camera ?c2)
    (plate ?pl) (time ?t2))
  (test (> ?t2 ?t1))
  (Camera (id ?c1) (highway ?hw)
    (direction ?d) (position ?p1))
  (Camera (id ?c2) (highway ?hw)
    (direction ?d) (position ?p2))
  (test (> (speed ?t1 ?p1 ?t2 ?p2)
           *speed-limit*))
=>
  (printout "Car too fast: "
            ?pl " at "
    (speed ?t1 ?p1 ?t2 ?p2) " km/h."))
```

Challenges:
Combining the Data

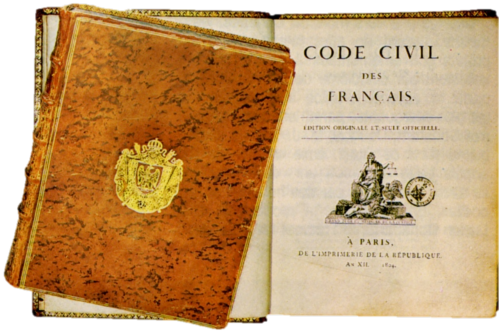
```
(defrule StationaryCar
  (CarInTraffic (camera ?c)
    (plate ?pl) (time ?t1))
  (CarInTraffic (camera ?c)
    (plate ?pl) (time ?t2))
  (test (> (- ?t2 ?t1) 30))
  (Camera (id ?c) (highway ?hw)
    (direction ?d) (position ?p))
=>
  (assert (StationaryCar (camera ?c)
             (plate ?pl) (time ?t1))))
```

```
(defrule StationaryCars
  (Camera (id ?c) (highway ?hw)
    (direction ?d) (position ?p))
  (StationaryCar (camera ?c)
    (plate ?pl1) (time ?t1))
  (StationaryCar (camera ?c)
    (plate ?pl2) (time ?t2))
  (StationaryCar (camera ?c)
    (plate ?pl3) (time ?t3))

  (test (!= ?pl1 ?pl2 ?pl3))
  (test (< 0 (- ?t2 ?t1) 1))
  (test (< 0 (- ?t3 ?t2) 1))
=>
  (assert (StationaryCars
             (camera ?c) (time ?t1))))
```

How to approach such scenarios?

- Declarative rules!
- Online processing of real-time events!

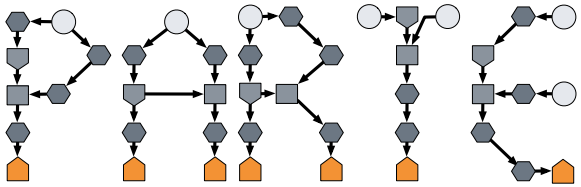


1 set of rules
1 set of facts
(transparent distribution)

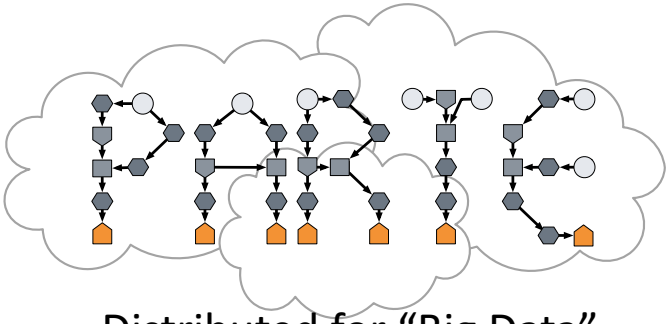
Load balancing

Elasticity

Our Approach



Parallel + Soft Real-Time

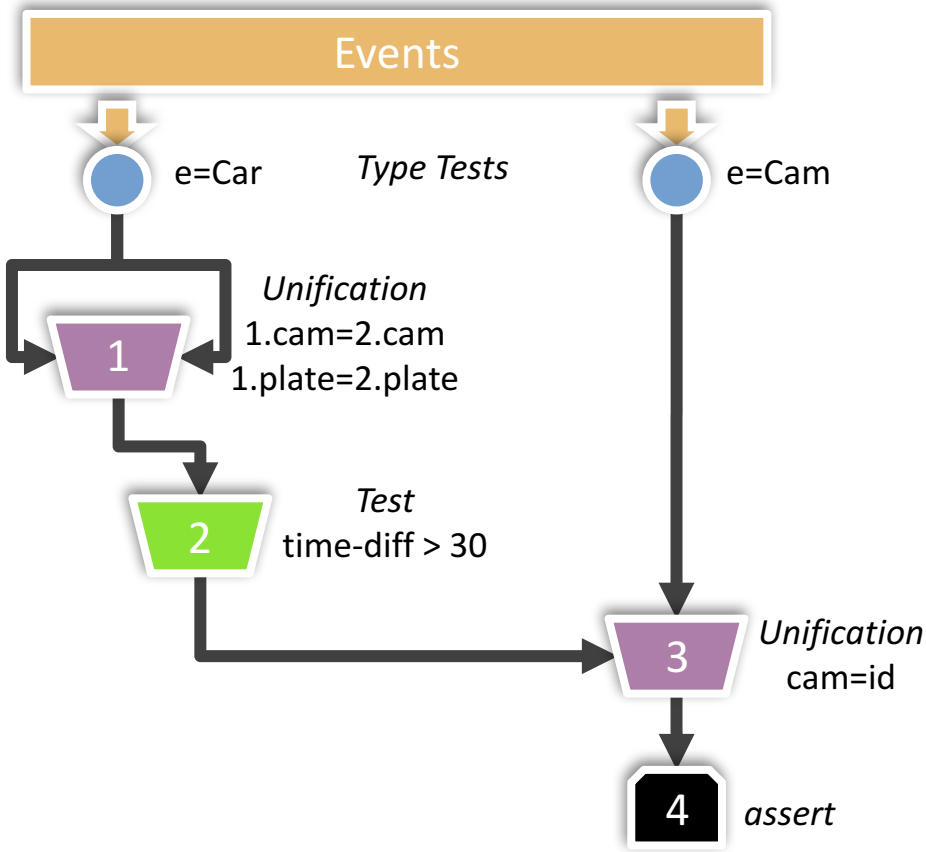


Distributed for “Big Data”

- Declarative rules
- Mobile actors
- Central managing interface
- Simple heuristics for load balancing

PARTE: A Parallel, Actor-based Rete Engine

```
(defrule StationaryCar
  (Car (cam ?c) (plate ?plt) (time ?t1))
  (Car (cam ?c) (plate ?plt) (time ?t2))
  (test (> (- ?t2 ?t1) 30))
  (Cam (id ?c) (position ?p))
=>
  (assert (StationaryCar (pos ?p)))
```

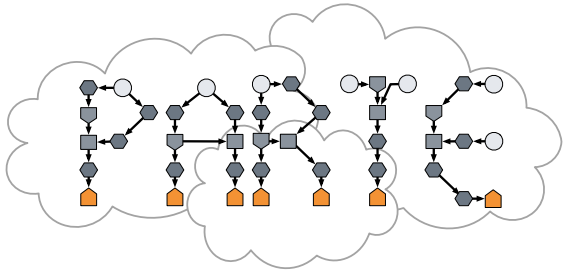


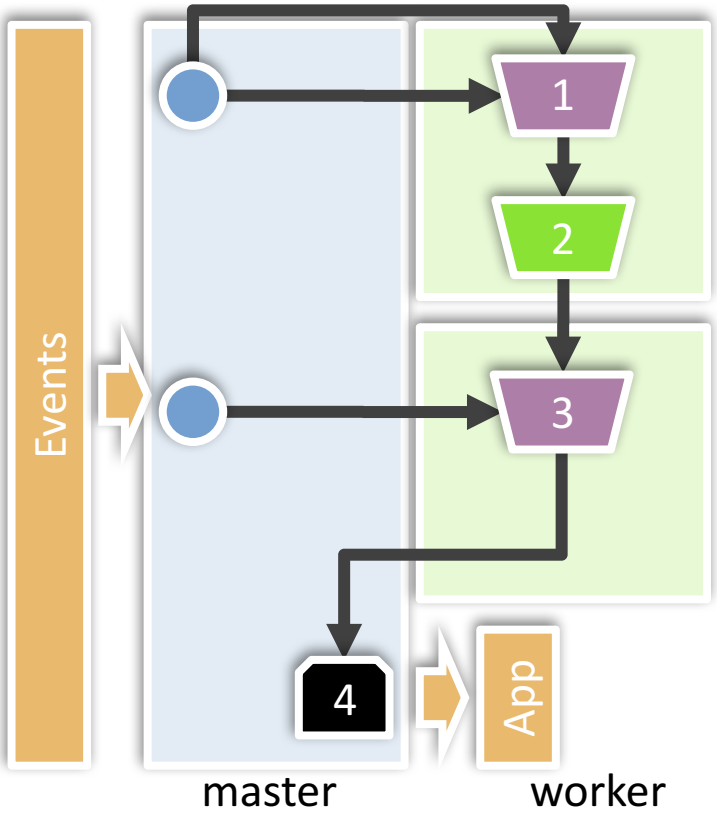
- Each node an actor
- Unification memory intensive
- Tests computationally intensive

[1] Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem, Charles L. Forgy. Artif. Intell. 19(1):17–37 (1982)
[2] Parallel Gesture Recognition with Soft Real-Time Guarantees, T. Renaux, L. Hoste, S. Marr, and W. De Meuter. AGERE'12, 35–46.

Cloud PARTE

From Parallel to Distributed



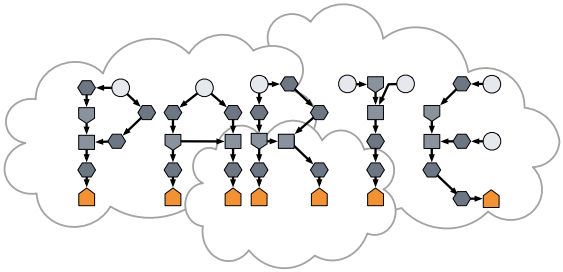


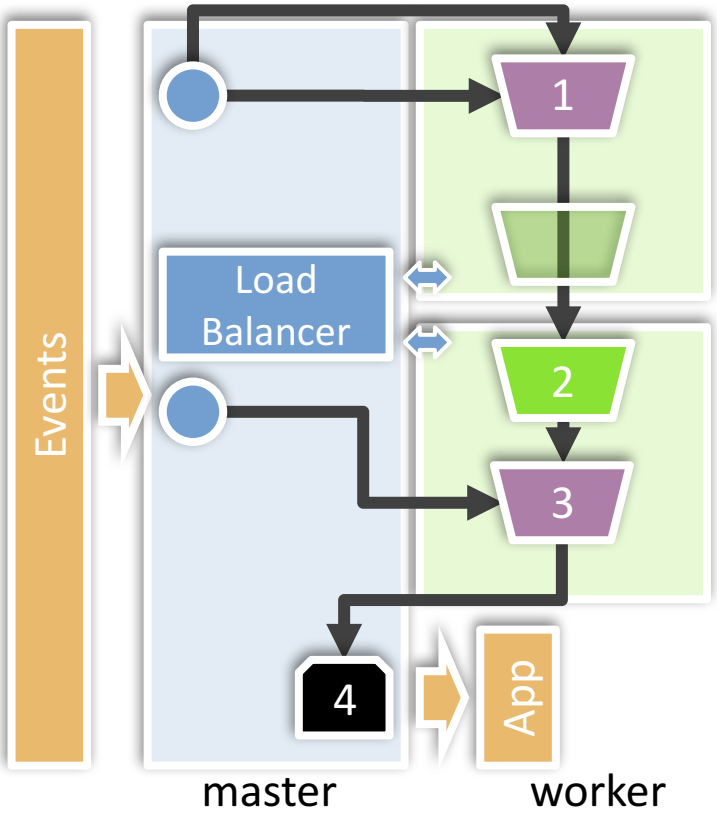

```
(defrule StationaryCar
  (Car (cam ?c) (plate ?plt) (time
?t1))
  (Car (cam ?c) (plate ?plt) (time
?t2))
  (test (> (- ?t2 ?t1) 30))
  (Cam (id ?c) (position ?p))
=>
  (assert (StationaryCar (pos ?p)))
```

- Transparent Distribution

Cloud PARTE

From Parallel to Distributed





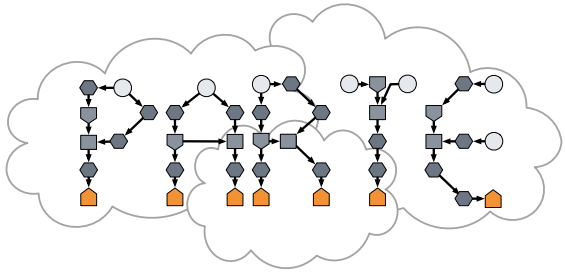

```
(defrule StationaryCar
  (Car (cam ?c) (plate ?plt) (time
?t1))
  (Car (cam ?c) (plate ?plt) (time
?t2))
  (test (> (- ?t2 ?t1) 30))
  (Cam (id ?c) (position ?p))
=>
  (assert (StationaryCar (pos ?p)))
```

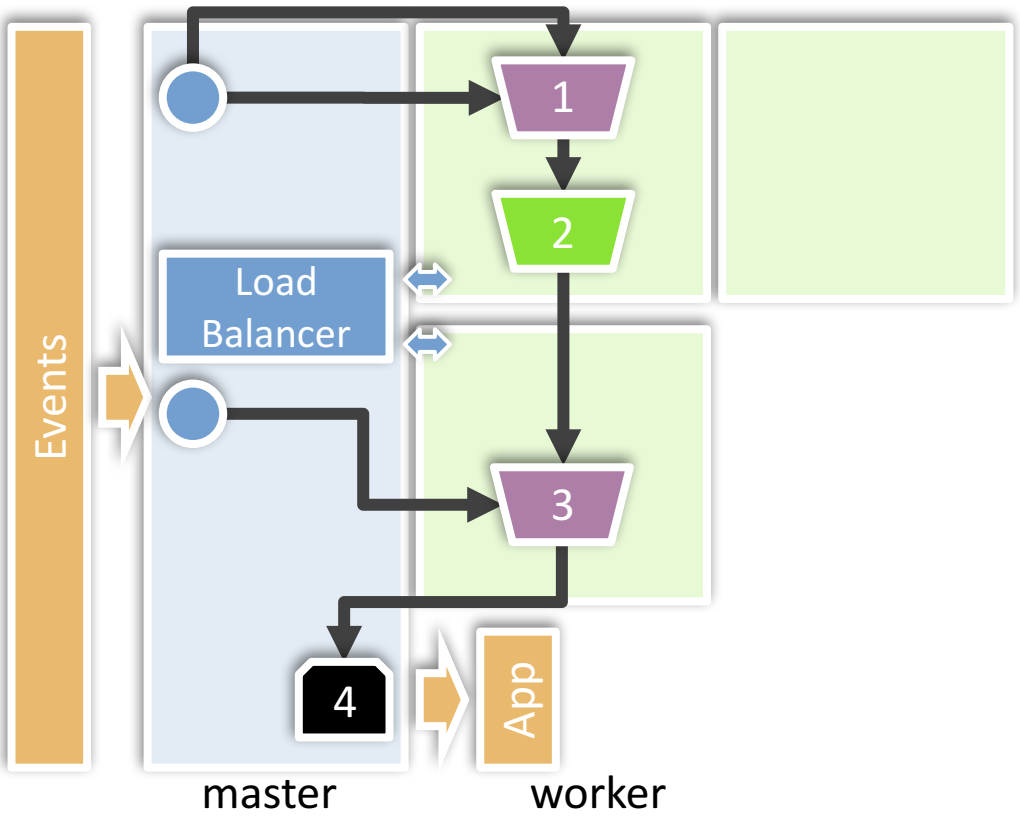
- Automatic Load Balancing
- Mobile actors
- Heuristic
  - Length of message queue

Cloud PARTE builds on Theron (C++ actor library, http://www.theron-library.com/)

Cloud PARTE

From Parallel to Distributed

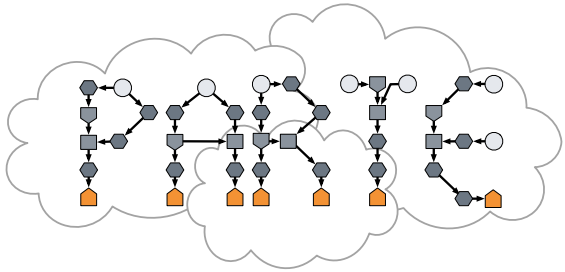


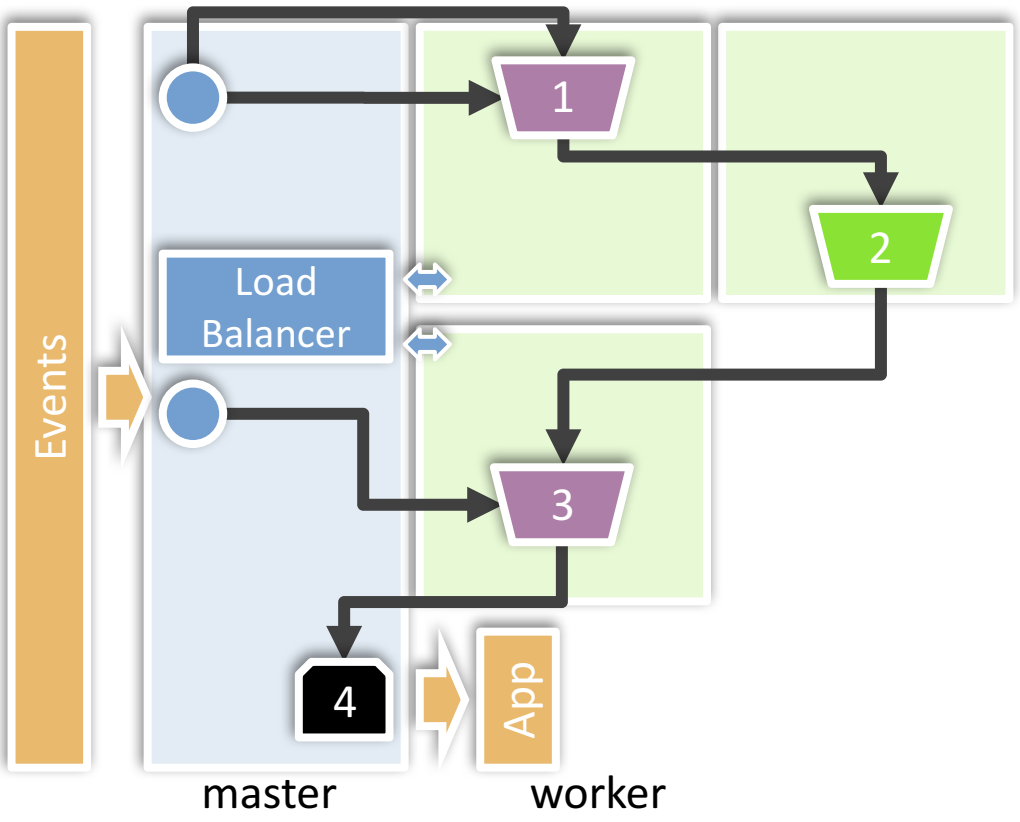


- Elasticity
  - Adding/removing of workers
- Coordination required (currently)

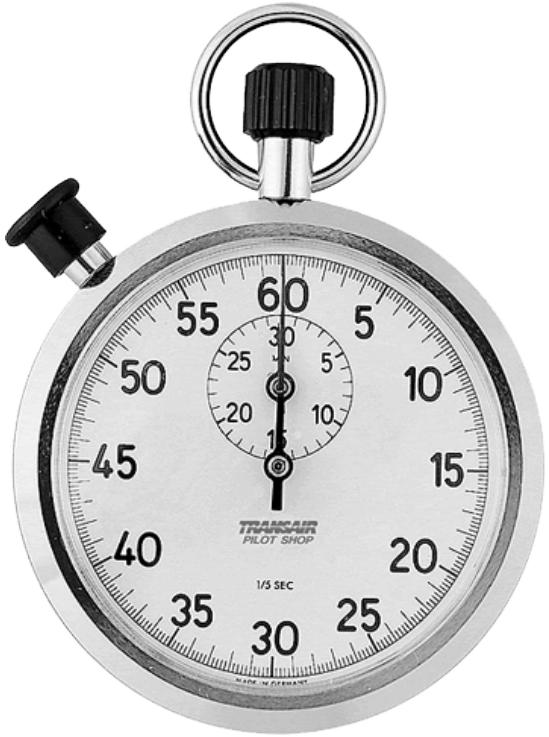
Cloud PARTE

From Parallel to Distributed





- Elasticity
  - Adding/removing of workers
- Coordination required (currently)

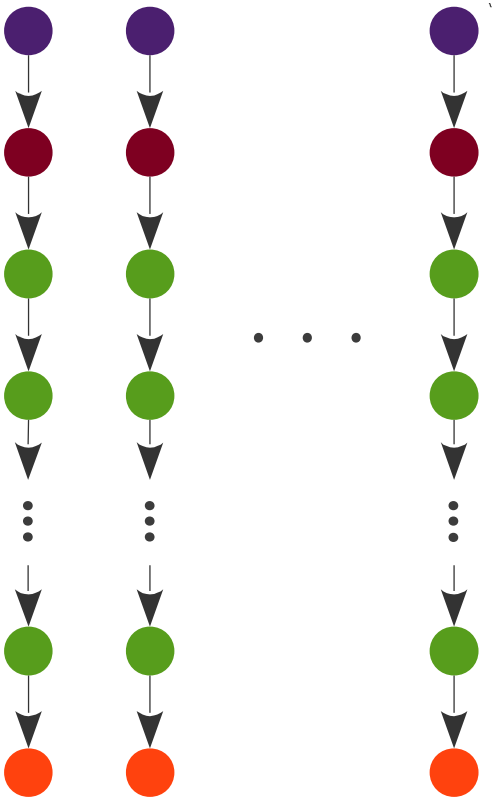


Performance

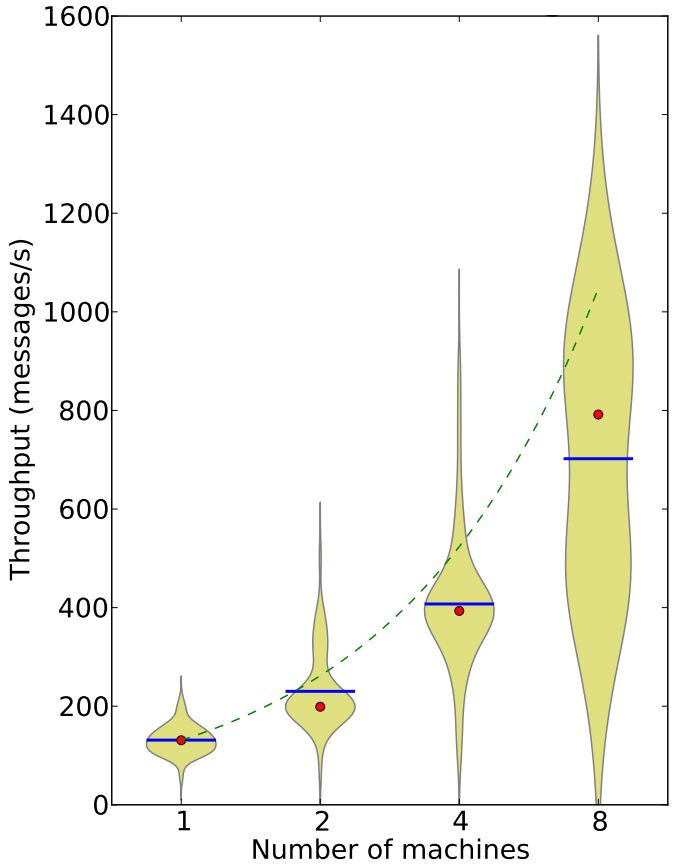
EVALUATION

Microbenchmarks

16 heavy tests



Cloud PARTE Scalability

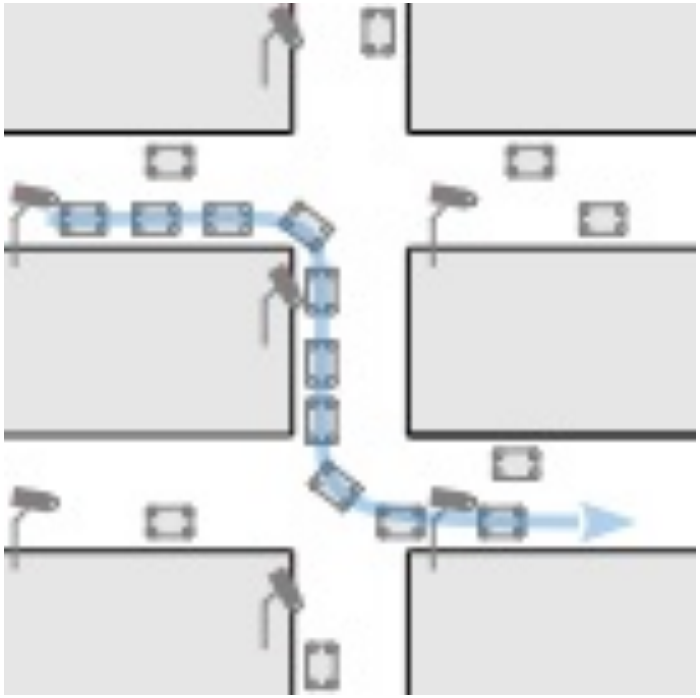


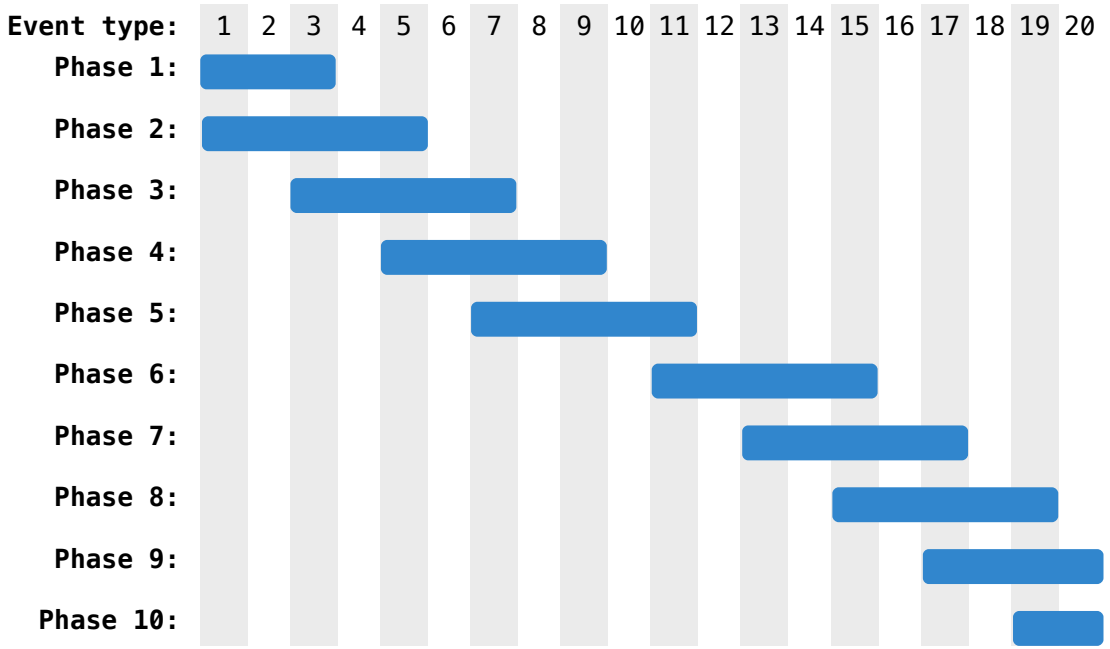
PARTE vs. CloudPARTE	Slow down
101 simple tests	18x
501 simple tests	29x
101 complex tests	5x
… with variables	5x
10x101 simple tests	60x
10x8 heavy tests	1x
16 heavy tests	4x
32 heavy tests	1x
64 heavy tests	4x
128 heavy tests	3x
Joining tests	6x
Search	2x

Communication overhead

Single machine

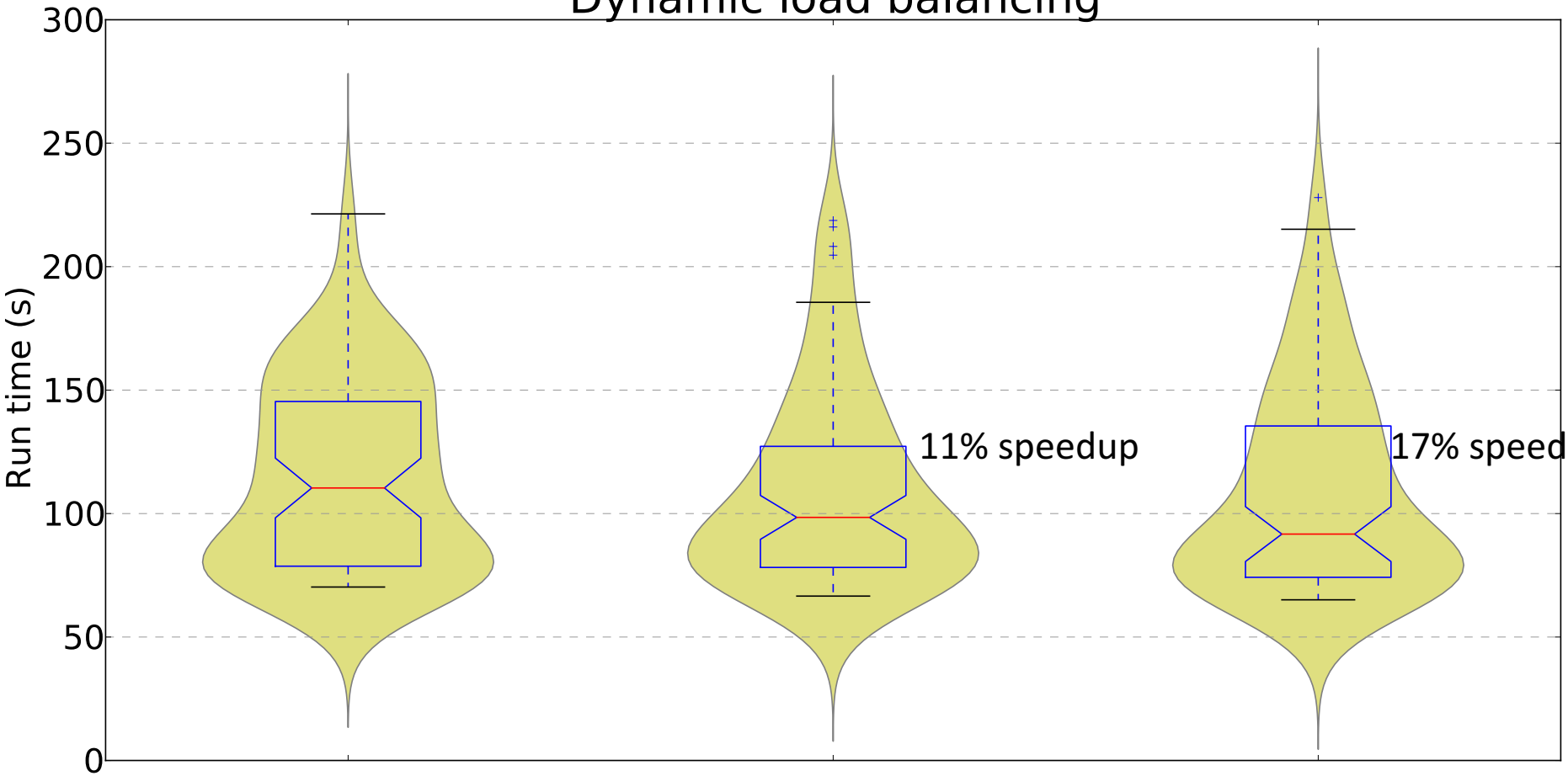
Performance: Traffic Scenario





Simulating events from variety of sources

Dynamic load balancing

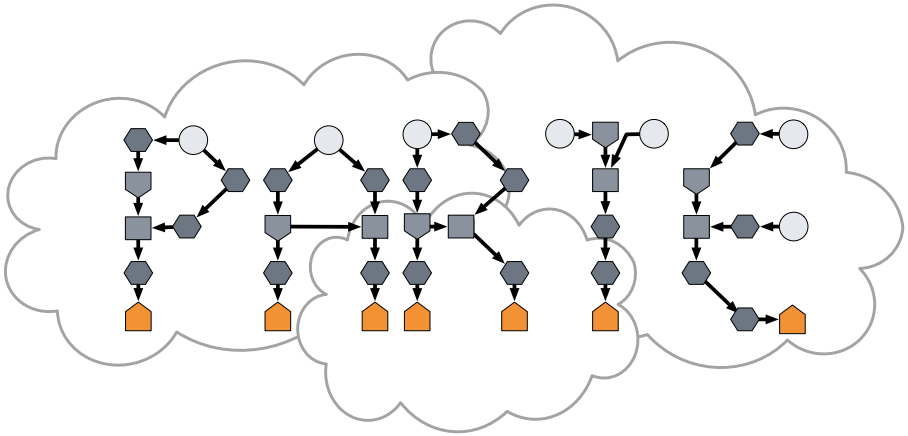


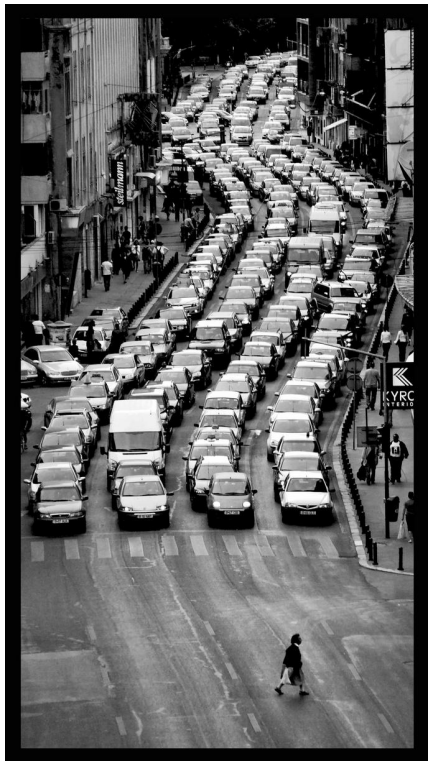
2 machines
Static Actor Distribution

2 machines
Static Actor Distribution +
Dynamic Load Balancing

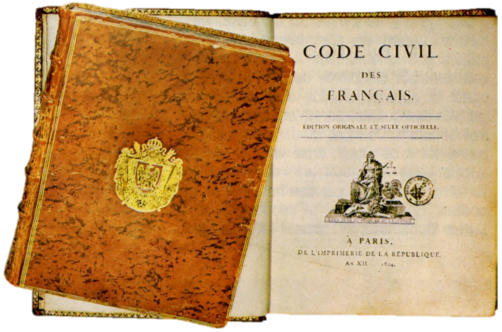
3 machines
Static Actor Distribution +
Dynamic Load Balancing +
Empty Machine

CONCLUSION





Traffic Management



1 set of rules
1 set of facts
(transparent distribution)

Load balancing

Elasticity