A MapReduce-based architecture for rule matching in production system

Bin Cao

2010.12.1

Agenda

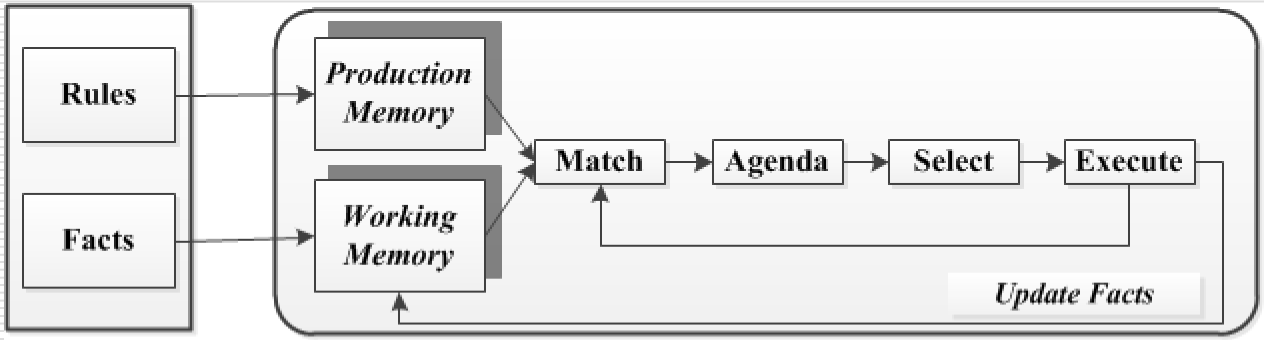
- Introduction
- Related Work
- Architecture
- Definition
- Implementation
- Experimental evaluation
- Conclusion and future work

Agenda

- **Introduction**
- Related Work
- Architecture
- Definition
- Implementation
- Experimental evaluation
- Conclusion and future work

Introduction

- Business rules can improve people's business by providing a level of agility and flexibility.
- Production system (rule engine)



The mechanism of a production system

Introduction

- Most of the processing time is consumed by matching
- The efficiency will drop with the increase of rules and facts.
- Rete algorithm and its improvements
- **But,** the limitation will not disappear because of the bounded capability of one single computer.

Introduction

- MapReduce programming model
- To perform Rete concurrently in different computer

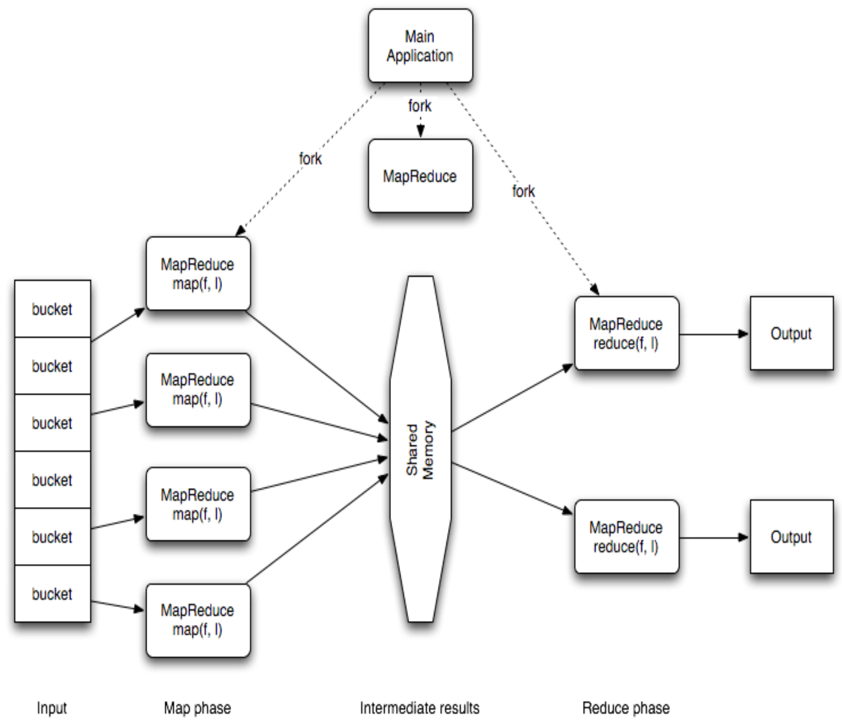


Figure 1

Agenda

- Introduction
- **Related Work**
- Architecture
- Definition
- Implementation
- Experimental evaluation
- Conclusion and future work

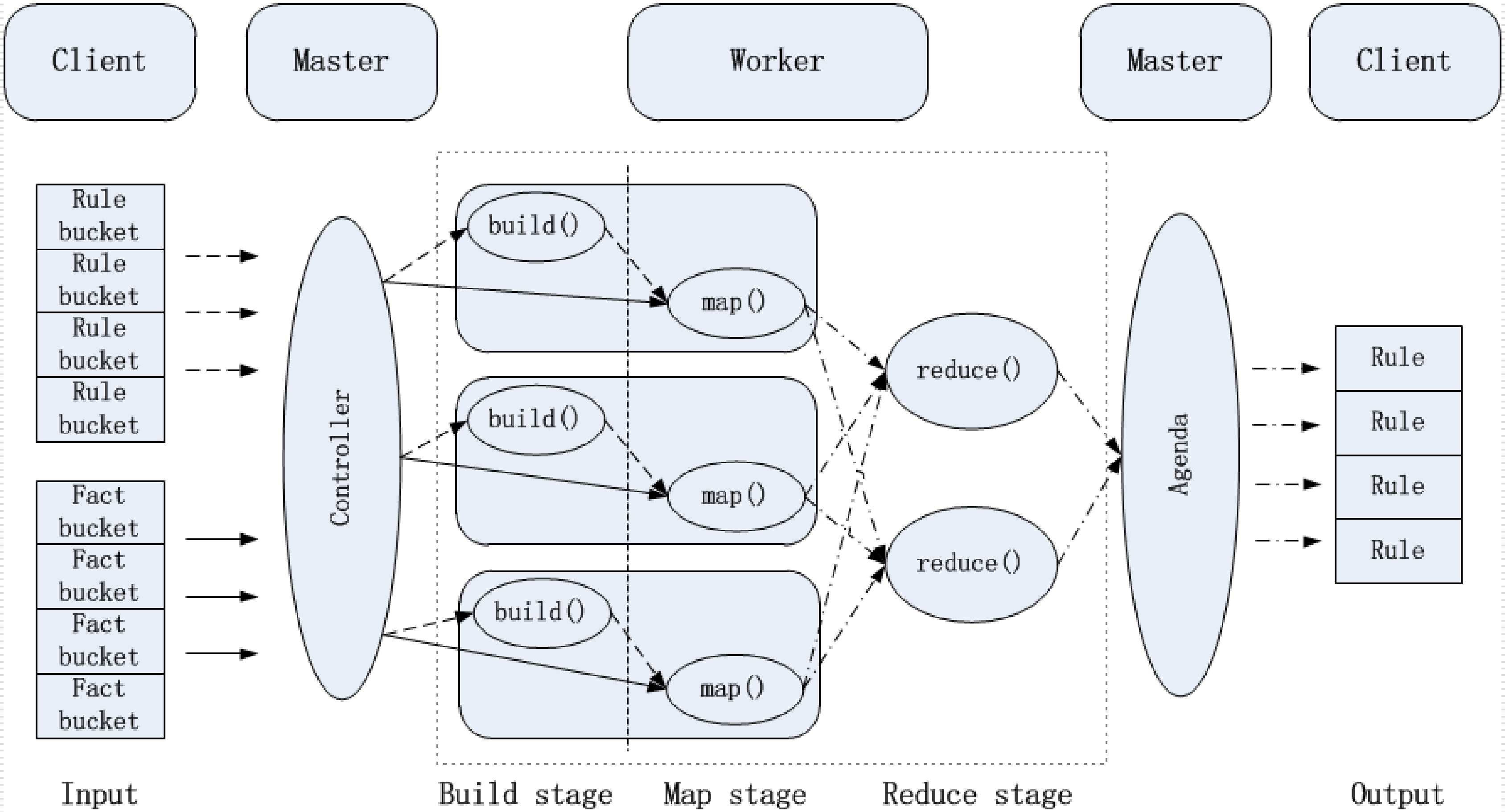
Related Work

- Parallel **firing** of rules
  - *Toru Ishida. Parallel, Distributed and Multi-Agent Production Systems.*
- Parallel but **not distributed**
  - *Anoop Gupta, Charles L.Forgy. Parallel OPS5 on the Encore Multimax*
- Parallel and distributed but **no Rete used**
  - *C. Wu, L. Lai, Y. Chang. Parallelizing CLIPS-based Expert Systems by the Permutation Feature of Pattern Matching.*

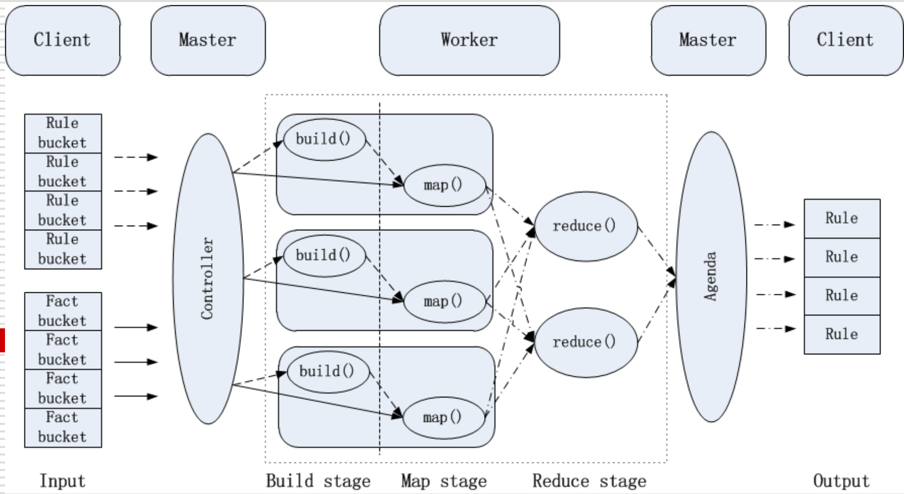
Agenda

- Introduction
- Related Work
- **Architecture**
- Definition
- Implementation
- Experimental evaluation
- Conclusion and future work

Architecture



Architecture



- Build stage
  - Rules are decomposed into sub-rules
  - Workers compile the sub-rules into a Rete net
- Map stage
  - Facts are passed to workers on demand
  - Facts will match with rules.
- Reduce stage
  - Reduce the results generated from map stage

Agenda

- Introduction
- Related Work
- Architecture
- **Definition**
- Implementation
- Experimental evaluation
- Conclusion and future work

Definition

If
…(*LHS*)
Then
…(*RHS*)

- **Definition 1 (Rule)**
  - *A rule, denoted **R**, is a tuple(LHS,RHS), where:*
    - *LHS is a finite set of **conditions** in a rule, called the **left hand side.***
    - *RHS is a finite set of **actions** in a rule, called the **right hand side.***
- **Definition 2 (Sub-rule)**
  - *Let **S** $\in$ LHS be a sub-rule of rule **R** **(Definition 1)**, iff, LHS belongs to **R**.*

Definition

Definition 3 (Rule base)

- *A matrix* $M_{(m,n,S)} = \begin{pmatrix} S_{<1,1>} & \cdots & S_{<1,n>} \\ \vdots & \ddots & \vdots \\ S_{<m,1>} & \cdots & S_{<m,n>} \end{pmatrix}$ *could be viewed as a rule base, where:*
  - ***m** represents for the number of rules.*
  - ***n** represents for the maximum number of sub-rules contained in each of rules above.*
  - ***S** represents for sub-rule (as defined in **Definition 2**)*

If we denote:

- $1 \leq \mathbf{r} \leq m$*: for rule ID in rule base **M**.*
- $1 \leq \mathbf{s} \leq n$*: for sub-rule ID in certain rule of rule base **M**.*

Then, $\mathbf{S}_{<\mathbf{r},\,\mathbf{s}>}$ *represents for sub-rule **S** identified by **s** in rule **R** identified by **r** in rule base **M***

Definition

- ***R_r** = (**$S_{<r, 1>}$ … $S_{<r, s>}$ … $S_{<r, n>}$**) shows that rule R, which identified by **r** in rule base, contains n sub-rules **$S_{<r, s>(1 \le s \le n)}$**. If the number of sub-rules in rule **R_r** is smaller than n, we equate the rest elements in **R_r** with null.*

Definition 4 (Firing paradigm)

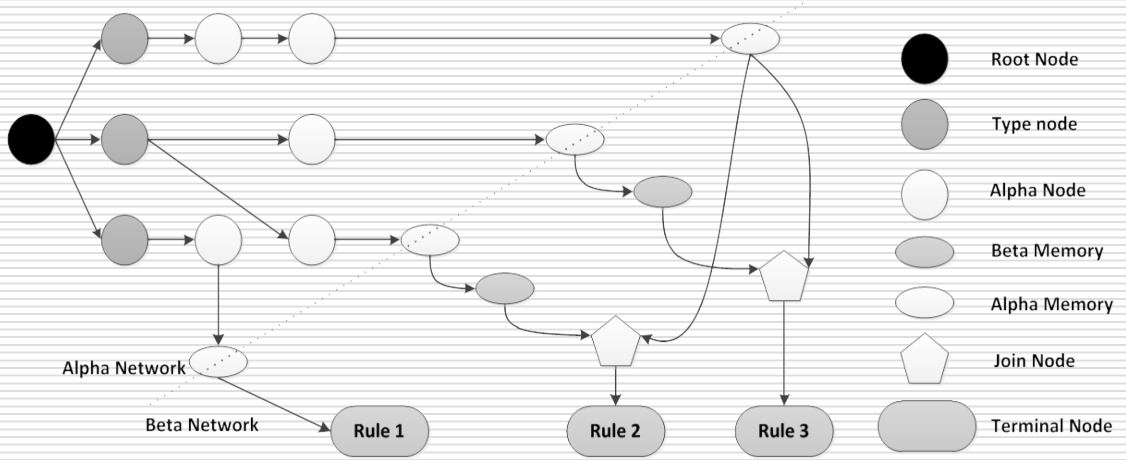
- *Two paradigms for firing **R_r** = (**$S_{<r, 1>}$ … $S_{<r, s>}$ … $S_{<r, n>}$**) are defined as following:*
  - ***AND:** .rule **R_r** can be fired if all the elements in **R_r** were matched simultaneously.*
  - ***OR:** rule **R_r** can be fired if a group of elements in **R_r** were matched simultaneously.*

Agenda

- Introduction
- Related Work
- Architecture
- Definition
- **Implementation**
- Experimental evaluation
- Conclusion and future work

Implementation

- **Build: preparations for rule matching**
  - Forming a rule base ***M***: decomposing rules into sub-rules
  - Distributing the sub-rules to different workers
  - Parsing sub-rules to a Rete-net



A Rete network

Implementation

- **Map: rule matching**

Function Map (Queue facts, List index_list)
{
/* Filter and match facts with Rete algorithm. */
matched_map (sub-rules_index, correspond_facts) = match_fact_with_Rete (facts, index_list) ;
/* According to the former definitions, classify and merge the matched_map by index.*/
classified_map (r, map(s, correspond_facts)) = classify_with_index (matched_map);
/* Save the result. * /
store (classified_map);
}

Implementation

- **Reduce:** responsible for correct transference of rule

```
Function  Reduce (RuleID r, List matched_subrule_list)
{
/* Classify and merge matched sub-rule list by s according to Definition 3.*/
merged_subrule_map (s, corresponding_subrule_list) = merge_with_s (matched_subrule_list);
/* Get firing paradigm of rule Rr.*/
switch (get_firing_paradigm (r)) //
  case AND
/* Judge whether the size of merged_subrule_map and sub-rule list of rule Rr in rule base is same. */
    if (equal (merged_subrule_map))
        transfer(r); // transfer the rule Rr to agenda of the master
  case OR
/* Judge whether there exists one or several group of elements in Rr was or were matched */
    if (exist_group_matched (r))
        transfer (r); // transfer the rule Rr to agenda of the master
}
```

Agenda

- Introduction
- Related Work
- Architecture
- Definition
- Implementation
- **Experimental evaluation**
- Conclusion and future work

Experimental Evaluation

- Goal: **to compare with sequence process**

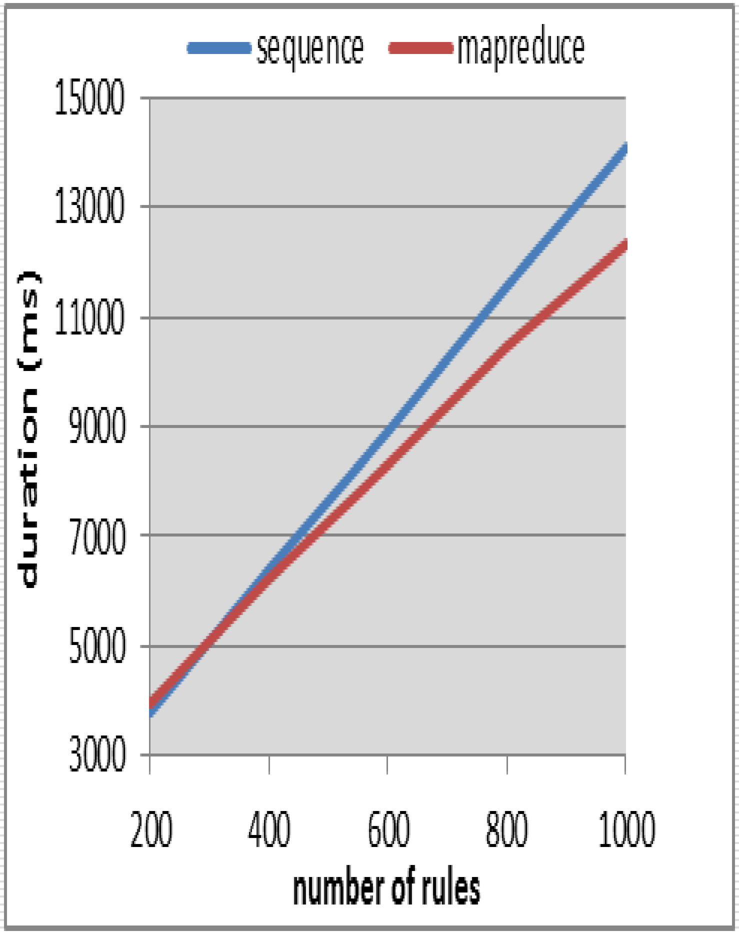
Master and Reduce Worker	
CPU	Intel Core2 Duo P8400@2.26GHz
Memory	3.00GB
HD	SATA 250GB
OS	Windows 7 Ultimate
Server	Apache-tomcat-6.0.18
LAN	100Mbps
Map Worker	
CPU	Intel Core2 Duo E7400@2.8GHz
Memory	2.00GB
HD	SATA 250GB
OS	Windows 7 Ultimate
Server	Apache-tomcat-6.0.18
LAN	100Mbps

The test environment configuration

Experimental Evaluation

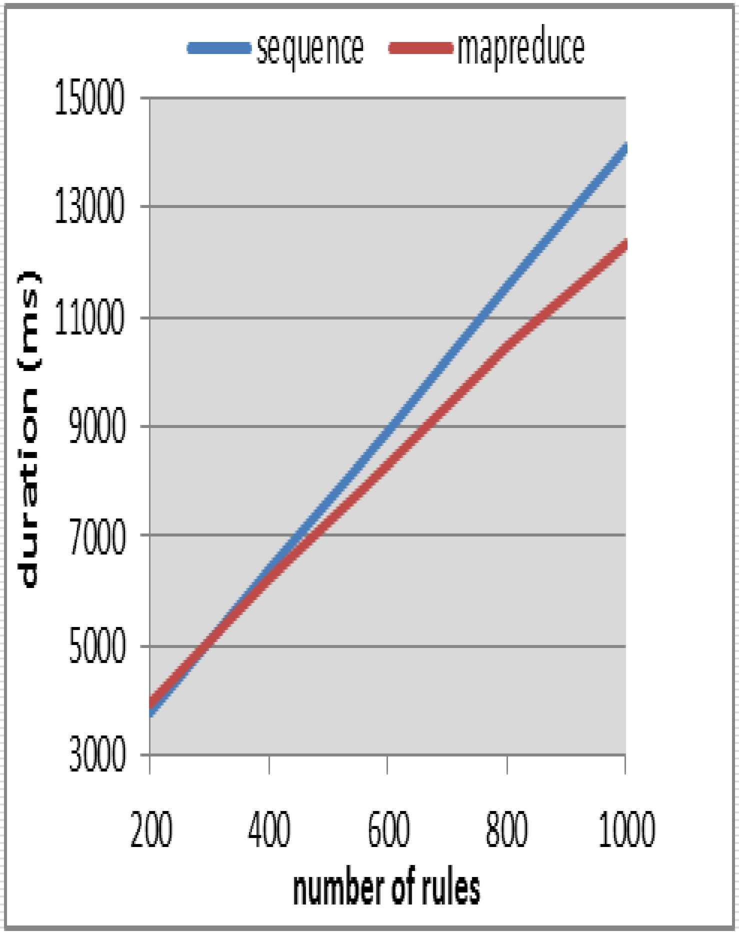
- At the bottom of the line, the MapReduce approach gains a little bit longer duration.

- Maybe the network transmission of matched result cost more time than matching process.



Experimental Evaluation

- As the number of rules increased, the gap between two lines is widening. The advantage of MapReduce approach appears more and more apparently.



Experimental Evaluation

- **Why the MapReduce approach does not double the performance?**
  - **The heavy network transmission.**
  - **Different load of each worker.**
  - **Different complexity of facts and rules.**
  - **…**

Experimental Evaluation

- **Nevertheless, the general trend is obvious:**

MapReduce process gains a less duration than sequential process when given the same number of rules, and with the increasing number of rules the MapReduce approach shows more efficient.

Agenda

- Introduction
- Related Work
- Architecture
- Definition
- Implementation
- Experimental evaluation
- **Conclusion and future work**

Conclusion and Future work

- Analysis coming from the relevant simulations confirm the efficiency of our architecture.
- To achieve better performance:
  - *How to compress the transferring data?*
  - *How to rescue from the dead or suspended worker?*
  - *How to utilize the parallel rules firing strategies?*
  - *...*

Thank You~

Bin Cao

2010.12.1