

A Multi-Agent Based Distributed Computing Platform for New Generation of EMS

Boming Zhang, Senior Member, IEEE

Chuanlin Zhao

Wenchuan Wu, Member, IEEE

Content

- ❖ **Introduction**
- ❖ **2-Level Multi-Agent Architecture**
 - Single Agent Architecture
 - Agent Society Architecture
- ❖ **Method to improve Real-time Performance**
 - Concurrency
 - Load Balance Strategy
- ❖ **Agent Coordination with 'Blackboard' Model**
- ❖ **Conclusion**

1 Introduction

Challenge in EPCC

- ❖ **N-EMS with 3-D Coordination** (2008 IEEE GM)
- ❖ **Heavy Computing Burden for N-EMS**
 - Spatial: entire grid model
 - Temporal: time scale; time evolution: historical, current and future
 - Objective: static, transient, voltage stability
 - Many cases
- ❖ **Reliability demand for On-Line App.**
 - Online, 7*24 usage
- ❖ **Flexibility demand**
 - Dynamically plugged in and out

Advantage

❖ Computer Hardware Improvement

- Multi-core: undertake more tasks simultaneously
- Cheap: form large-scale cluster
- Distributed and concurrent calculation environment

**Challenges raised
by modern
Power System**



**Advantages brought
by Hardware
improvement**

What should we do in software development to balance them?

Approach

❖ **Multi-Agent System (MAS)**

- methodology for modeling, designing and implementing large-scale, highly distributed and concurrent system

❖ **Achievement**

- Intelligent System Subcommittee of IEEE PES has founded a Working Group
- Plan to evolve EMS with MAS
- Limited to prototype currently, lack of MAS based systems in real field operation

2 Multi-Agent Architecture

Single Agent Architecture(1)

❖ Cognitive Agent

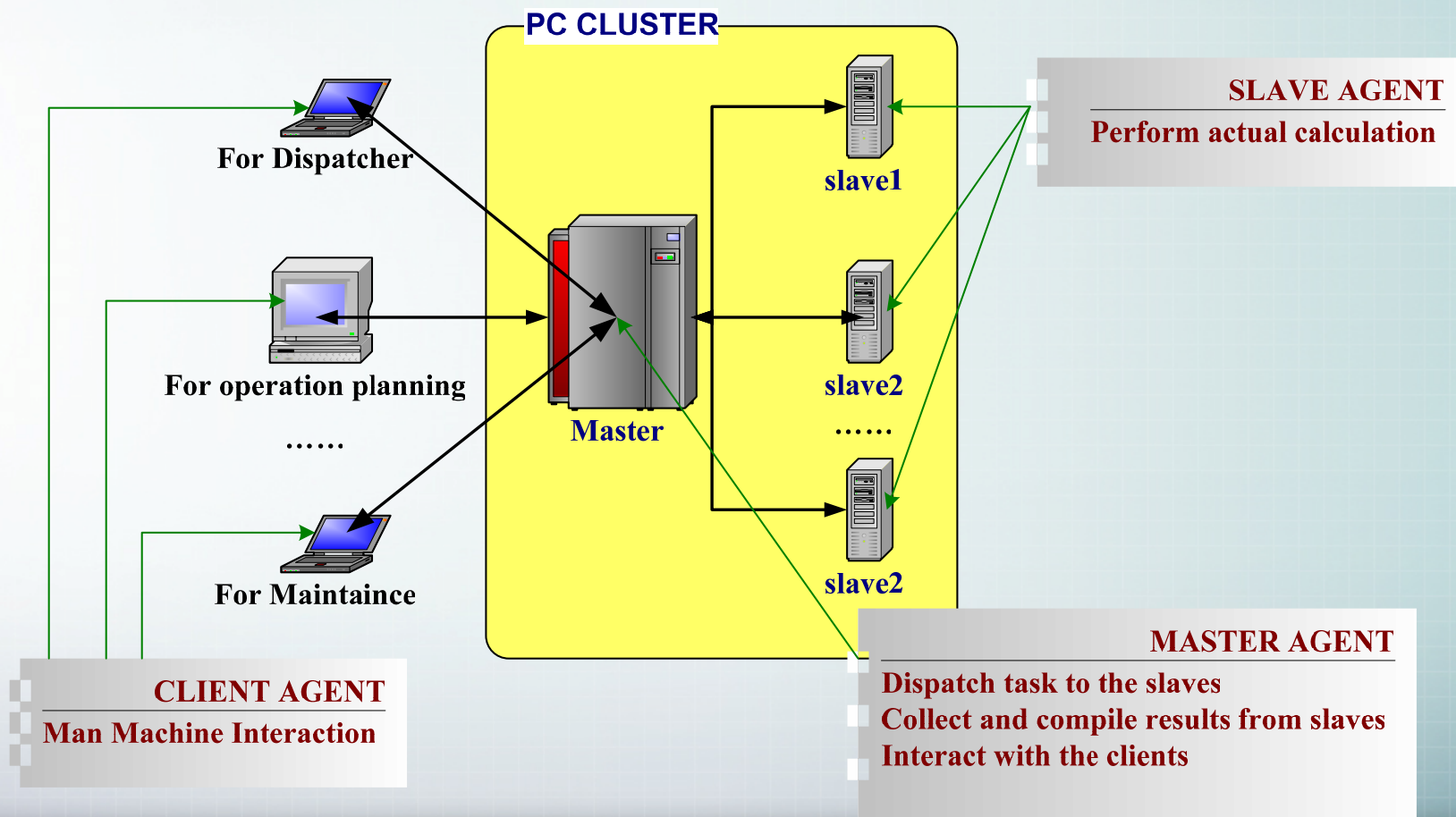
- Complex symbol reasoning
- Difficult to implement and operate
- Too slow for real-time usage

❖ Reactive Agent

- Whenever event detected, perform pre-defined act without any reasoning
- Fast enough for real-time usage

Single Agent Architecture(2)

❖ Reactive Agents are used in our platform



2-Level Federation for N-EMS (1)

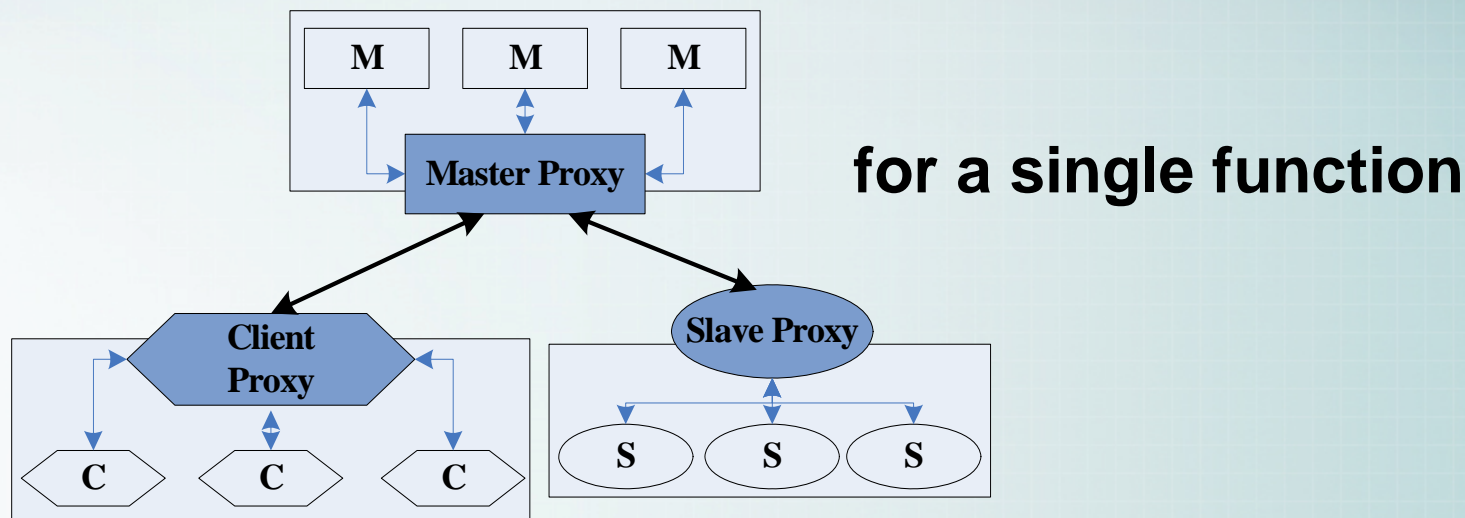
❖ Diversity of agent interaction requirements

- SSA, DSA, VSA, ...
- DSA: Contingency filtering, Detailed simulation on serious ones, Online TTC, ...
- Any Function needs: Client, Master and Slave Agents

❖ How To deal with the Complexity

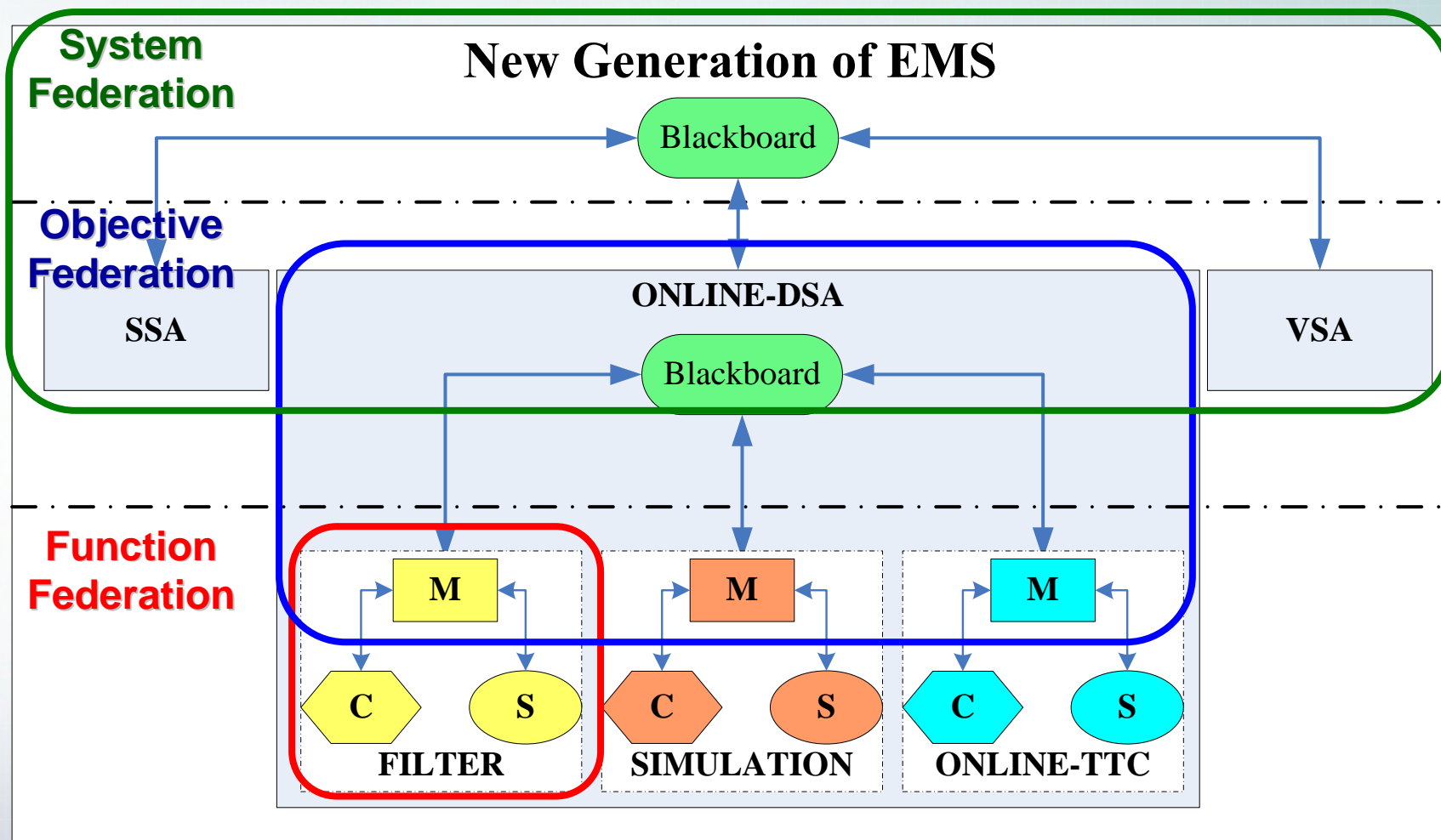
- ① Distributed interaction to fulfill a single function
- ② Coordination among Functions to build the whole system

Federation for Distributed Interaction



- ❖ Agents in one computer form a federation
- ❖ Proxies build "virtual links" among agents
- ❖ Single agent ignores communication details
- ❖ Decrease traffic flow on network

Federation for Function Coordination



3 Method to improve Real-time Performance

- Concurrency**
- Load Balance**

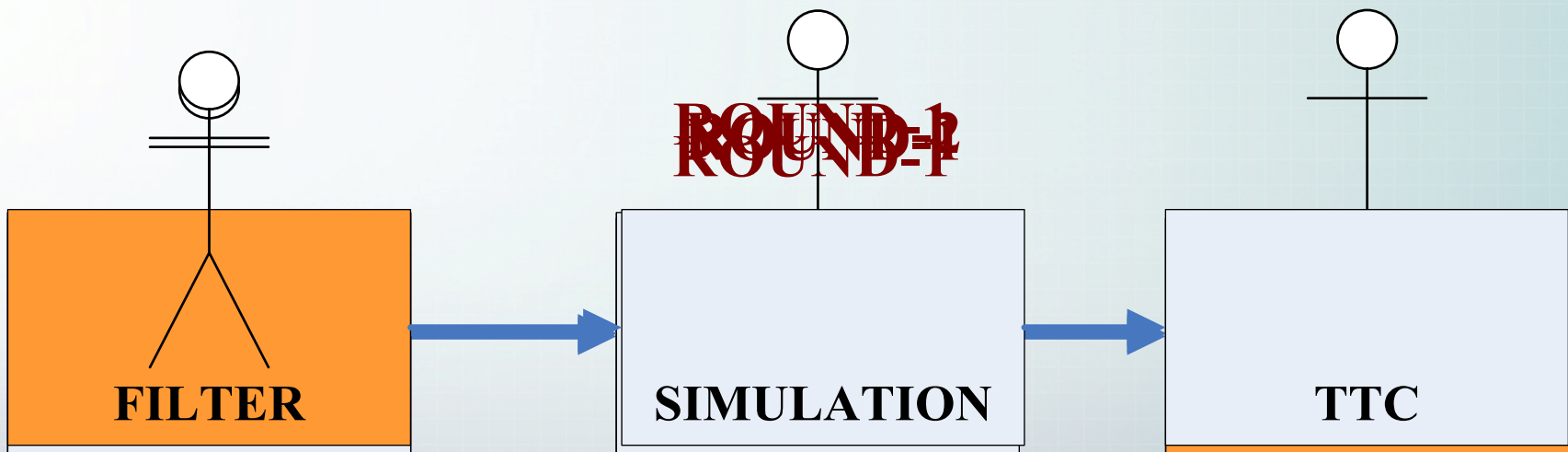
Concurrency

❖ Concurrency

- Inter-module concurrency
- Intra-module concurrency

❖ In C-EMS

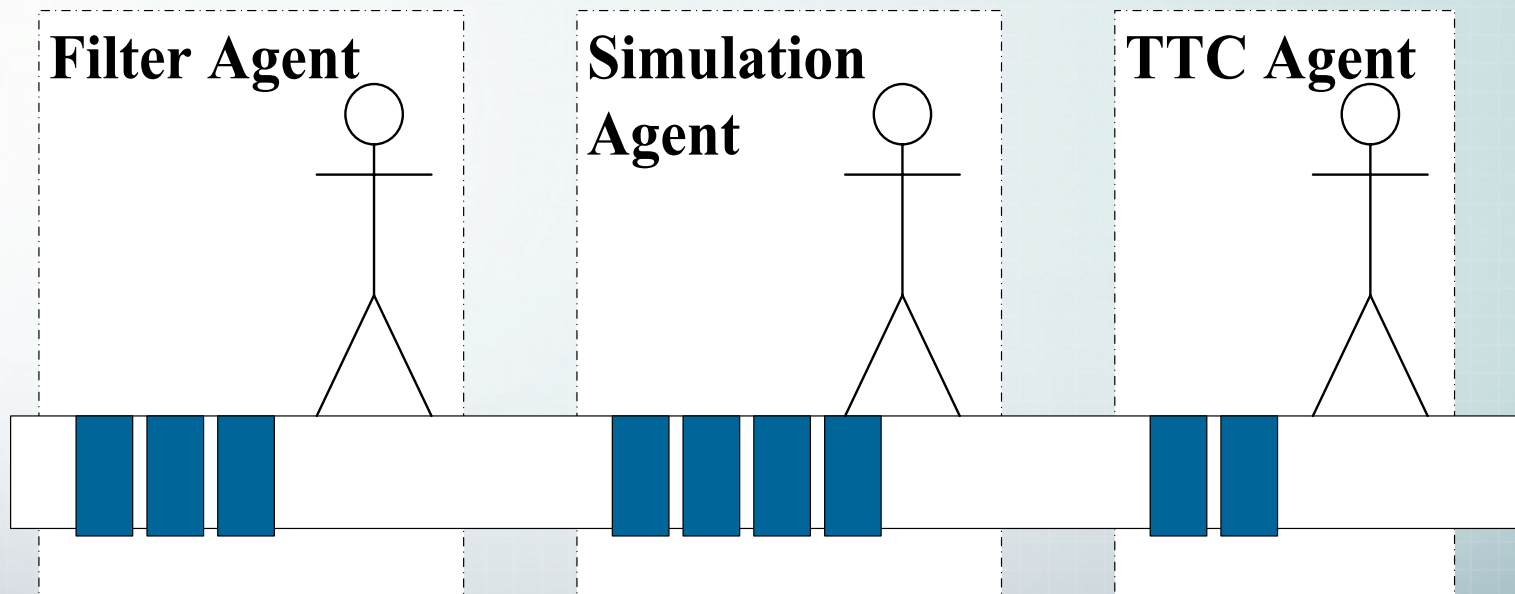
- Each module is implemented as subroutine or object
- Executed one by one



Inter-module Concurrency (1)

❖ Inter-module concurrency

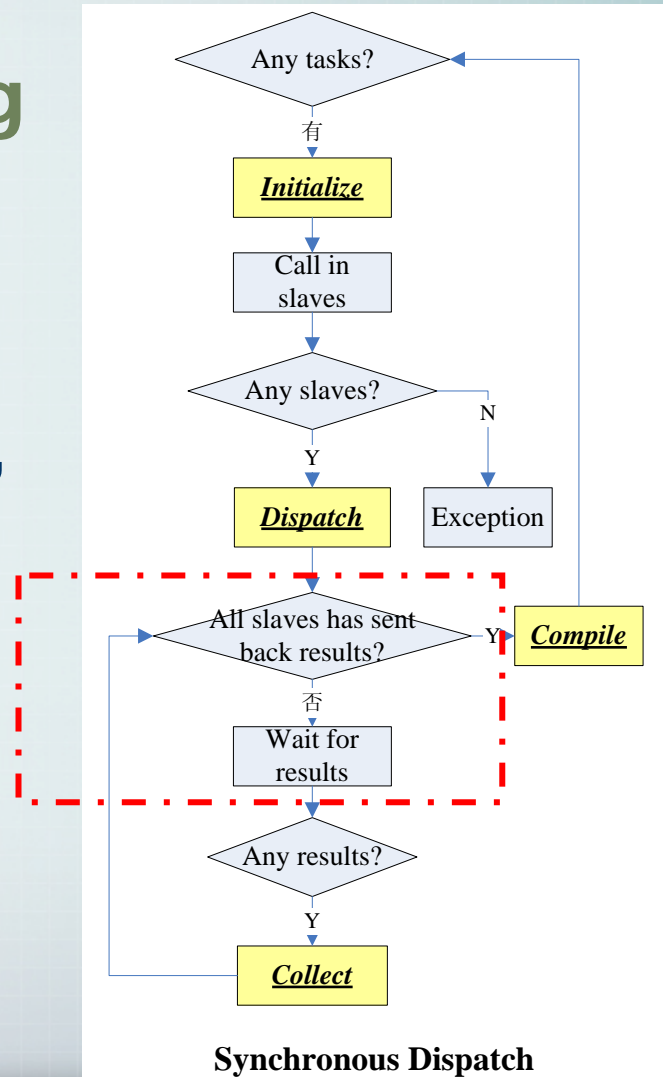
- Each agent runs on its own thread
- “Asynchronous messaging” replaces “synchronous method invocation”



Intra-module Concurrency (1)

1. Synchronous dispatching

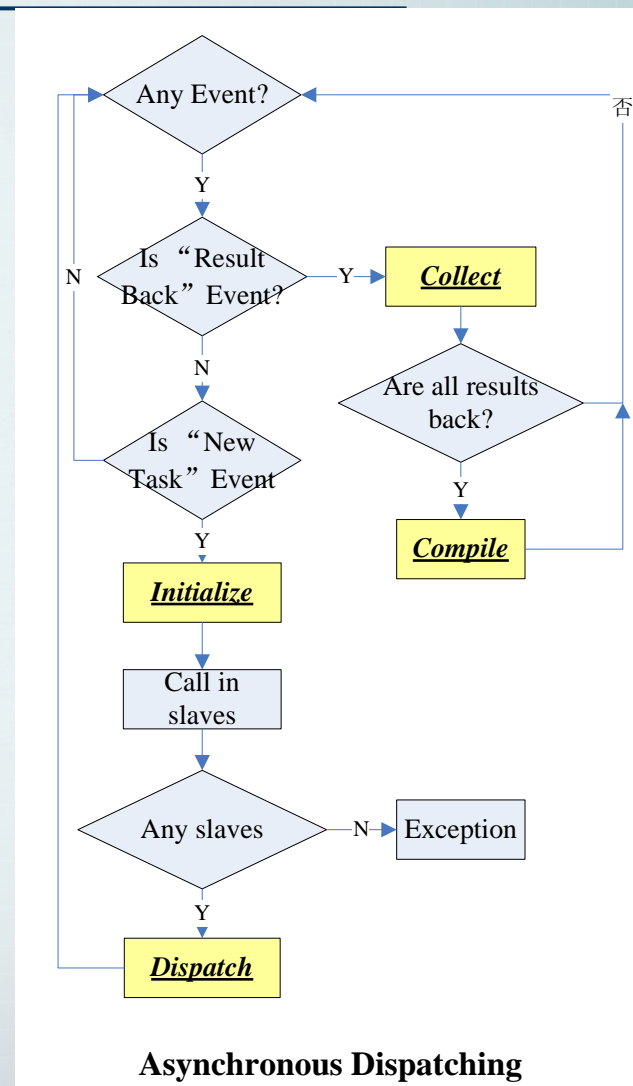
- After dispatching, block and wait for all slaves sending back results
- Serious “Short-Board Effect”



Intra-module Concurrency (2)

2. Asynchronous Dispatching

- After dispatching, dispatch the next task without waiting for all slaves sending back their results



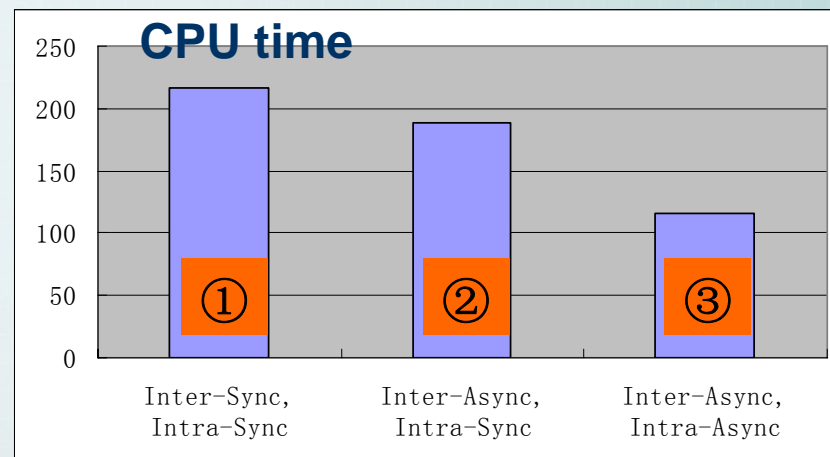
Concurrency Experiment

❖ Online-DSA Process

- Filtering → Detailed SBS Simulation → Online TTC

❖ Compare Three Ways

	Inter-Module	Intra-Module
①	sync	sync
②	async	sync
③	async	async



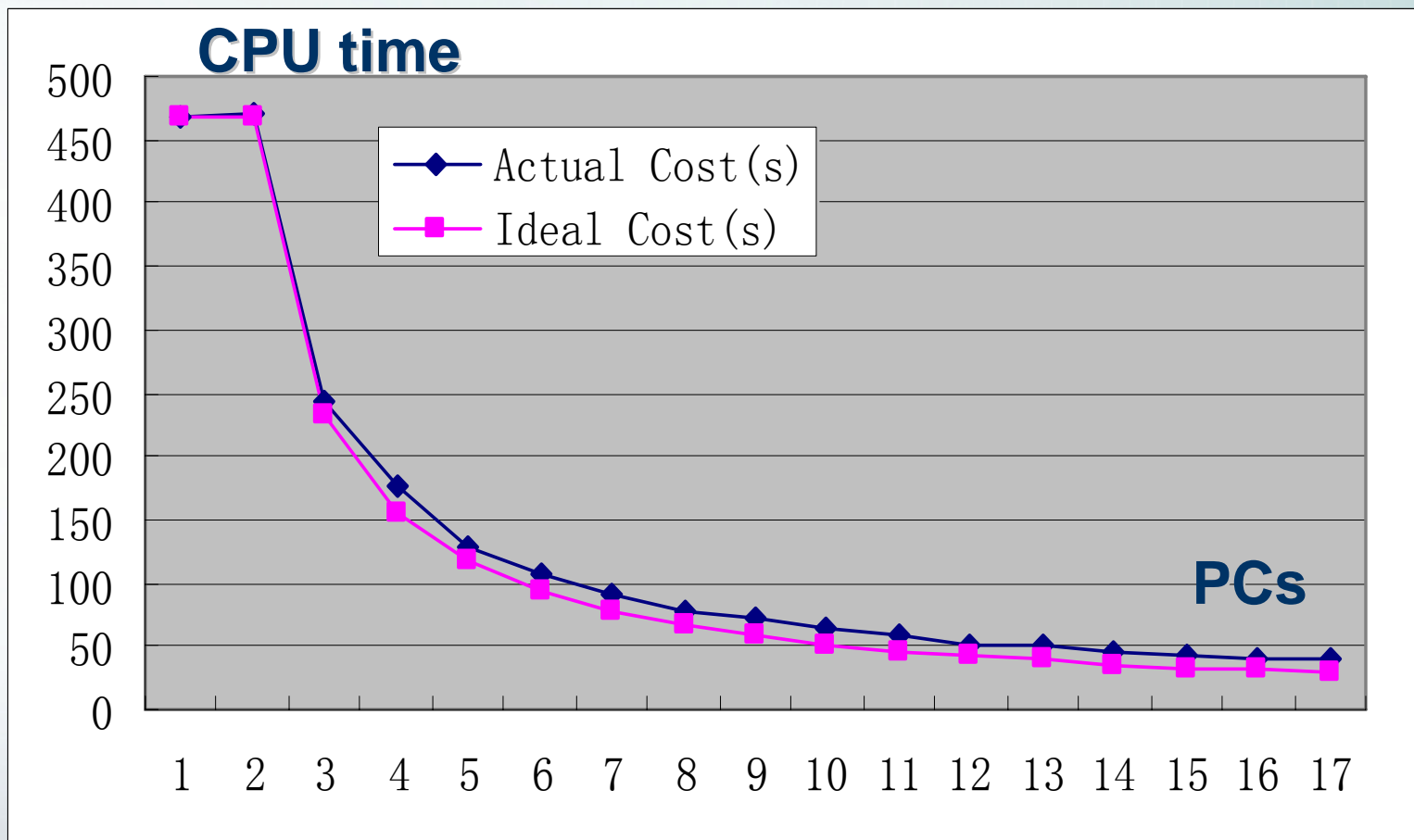
47% reduction

Load-Balance Strategy (1)

❖ Master-Driven Strategy (a push way)

- Each task expenses a similar cpu time
- The number of tasks is huge
- e.g. Online Filtering by ACS
- **Masters design the strategy, and push calculation load to each slave**

Load-Balance Strategy (2)



Load-Balance Strategy (3)

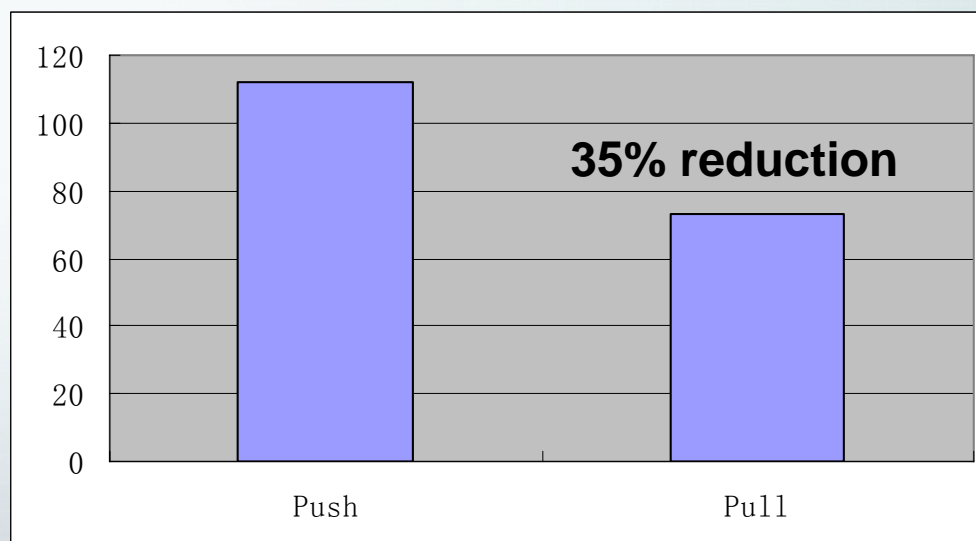
❖ Slave-Driven Strategy (a pull way)

- The number of tasks is relatively small
- Time cost for each task differs greatly and can not be predicted
- Master can not pre-design a proper strategy
- e.g. Online TTC Calculation
- **Slave pulls calculation load from master according to its capacity**

Load-Balance Strategy (4)

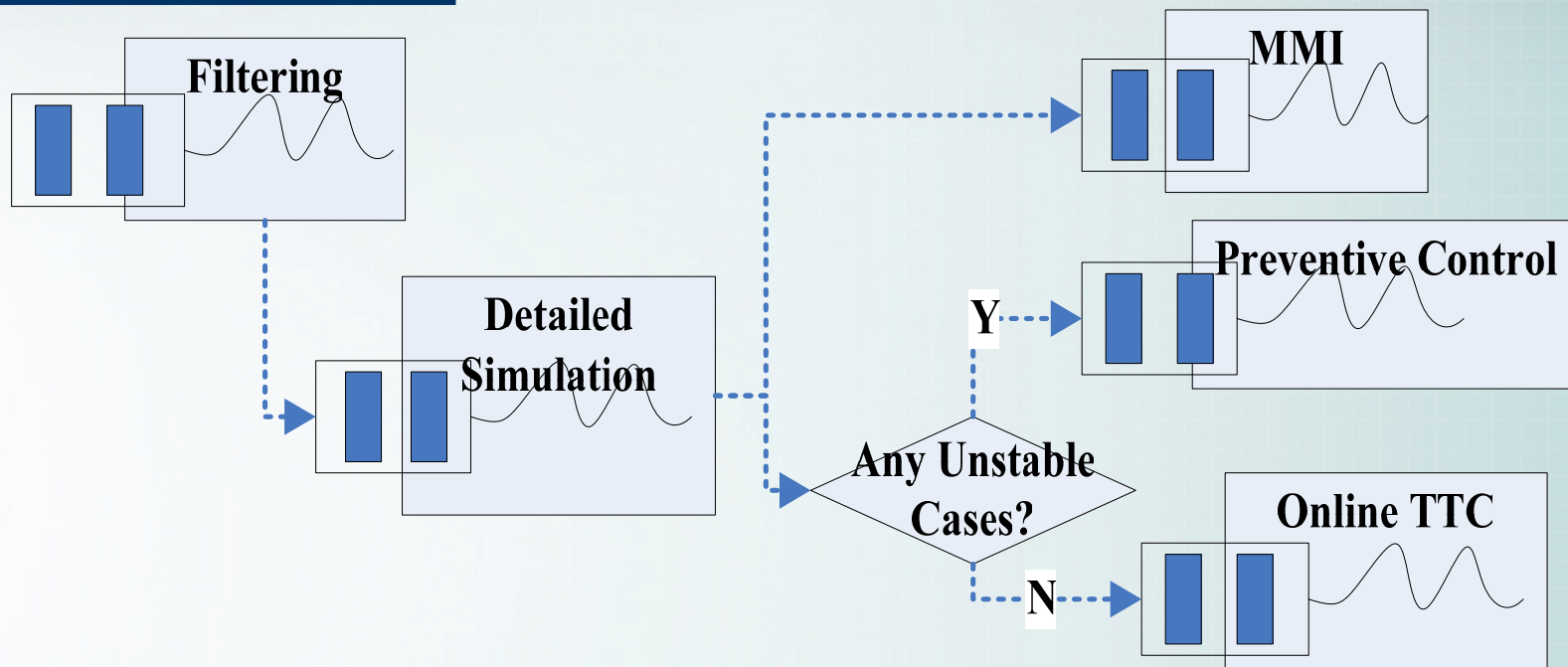
❖ Pull Strategy

- Each slave gets one task
- Light-load slave completes its work quickly, sends back the result, and pulls another task from master
- a slave may undertake only one heavy-load task
- Some others may undertake several light-load tasks



4 'Blackboard' Model is used to Coordinate Agents

Agent Coordination by 'Blackboard'



- ❖ Routing problem
- ❖ Blackboard is used

1. Original input
2. Intermediate solution
3. Final solution



Knowledge source (Agent)

An Expert on certain aspect of the overall problem

1. Read information on the blackboard
2. Process the information
3. and write results back onto the blackboard
4. Depends on notified



Controller

Monitor the blackboard

Decide which agent should be invoked next

Advantage of Blackboard

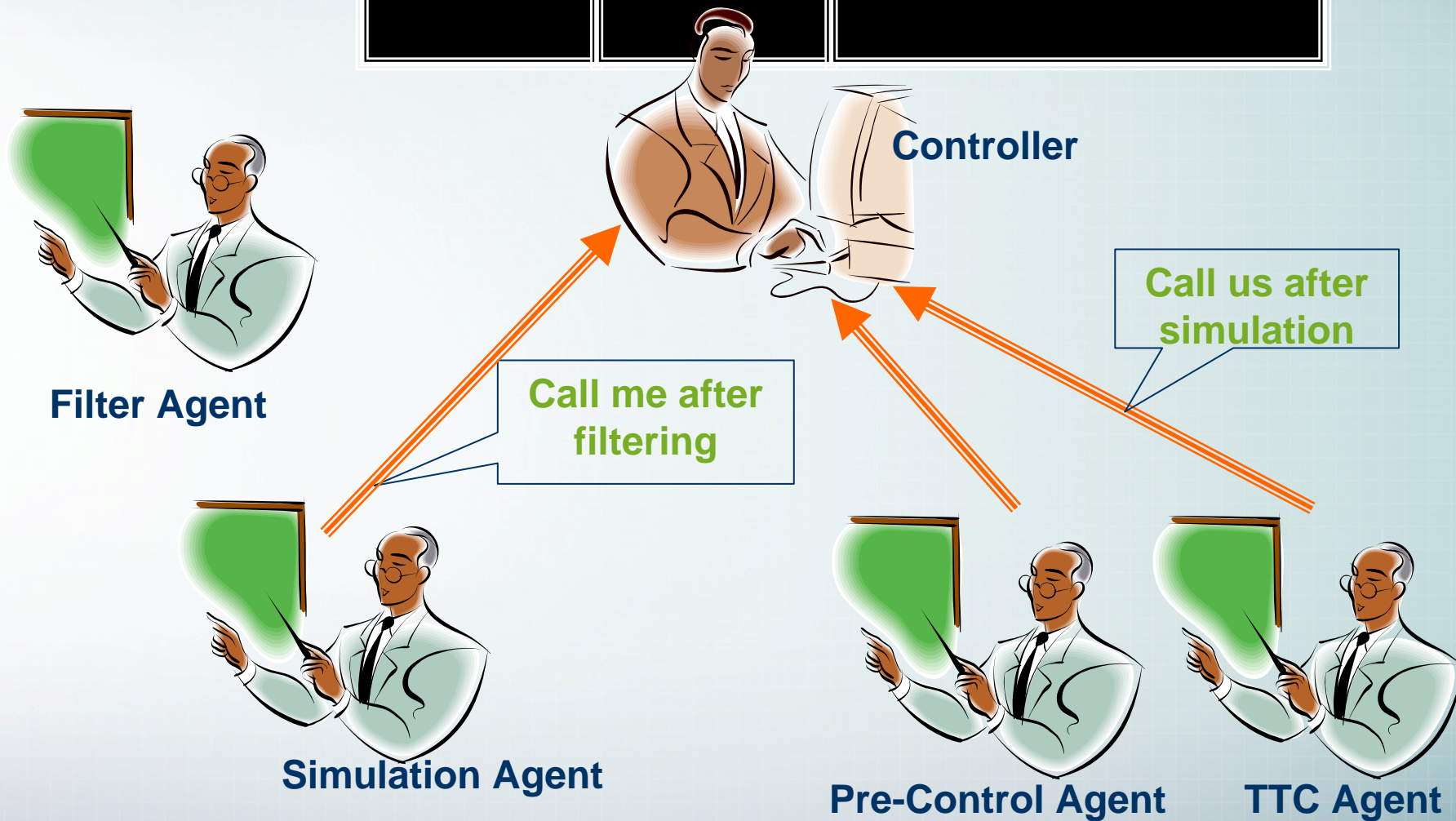
❖ Routing problem

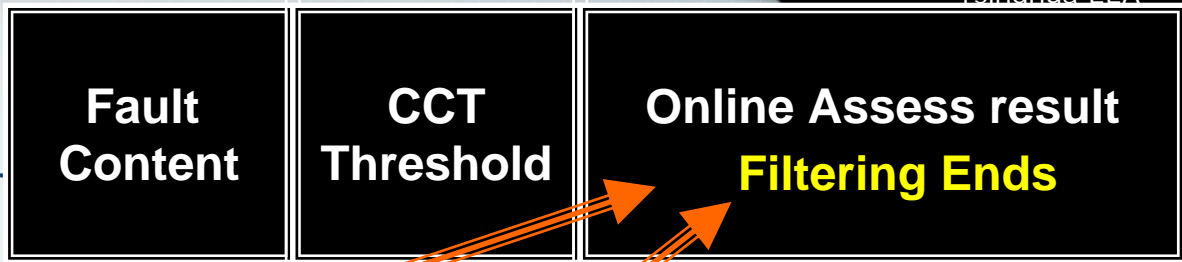
- After receiving a notice from controller, each agent queries further information, and then decides whether to act or not

❖ Pull Method

- Fetch necessary information from blackboard according to its own requirement

❖ Much more “autonomous”





1. I can calculate CCT with EEAC Algorithm, but the results may not be accurate enough



Filter Agent

2. I have observed this change



Controller

3. It's your turn



Simulation Agent



Pre-Control Agent



TTC Agent

Fault Content

CCT Threshold

Online Assess result
Simulation Ends



Filter Agent



Simulation Agent

1. I can calculate accurate CCT with your CCT as initial value

2. I've seen the change



Controller

3. Both of you, wake up please



Pre-Control Agent



TTC Agent

**Fault
Content**

**CCT
Threshold**

**Online Assess result
Simulation Ends**

I will calculate
TTC if no
unstable case is
detected



Filter Agent



Controller



Simulation Agent

I will design
remedy plan if
there are some
unstable cases



Pre-Control Agent



TTC Agent

5 Conclusion

- ❖ **2-level multi-agent architecture**
- ❖ **“concurrency” and “load-balance”technoques are used to improve Computing performance**
- ❖ **“Blackboard” mode is used to coordinate among agents**
- ❖ **The platform has been used to N-EMS**

THE END