Erlang @ SAP Research

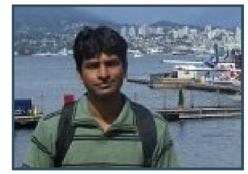
SYSTEMATIC THOUGHT LEADERSHIP FOR INNOVATIVE BUSINESS

Sumeet Bajaj, SAP Labs, Palo Alto

April 30, 2009

Erlangers @ SAP Research Palo Alto

Harald Weppner

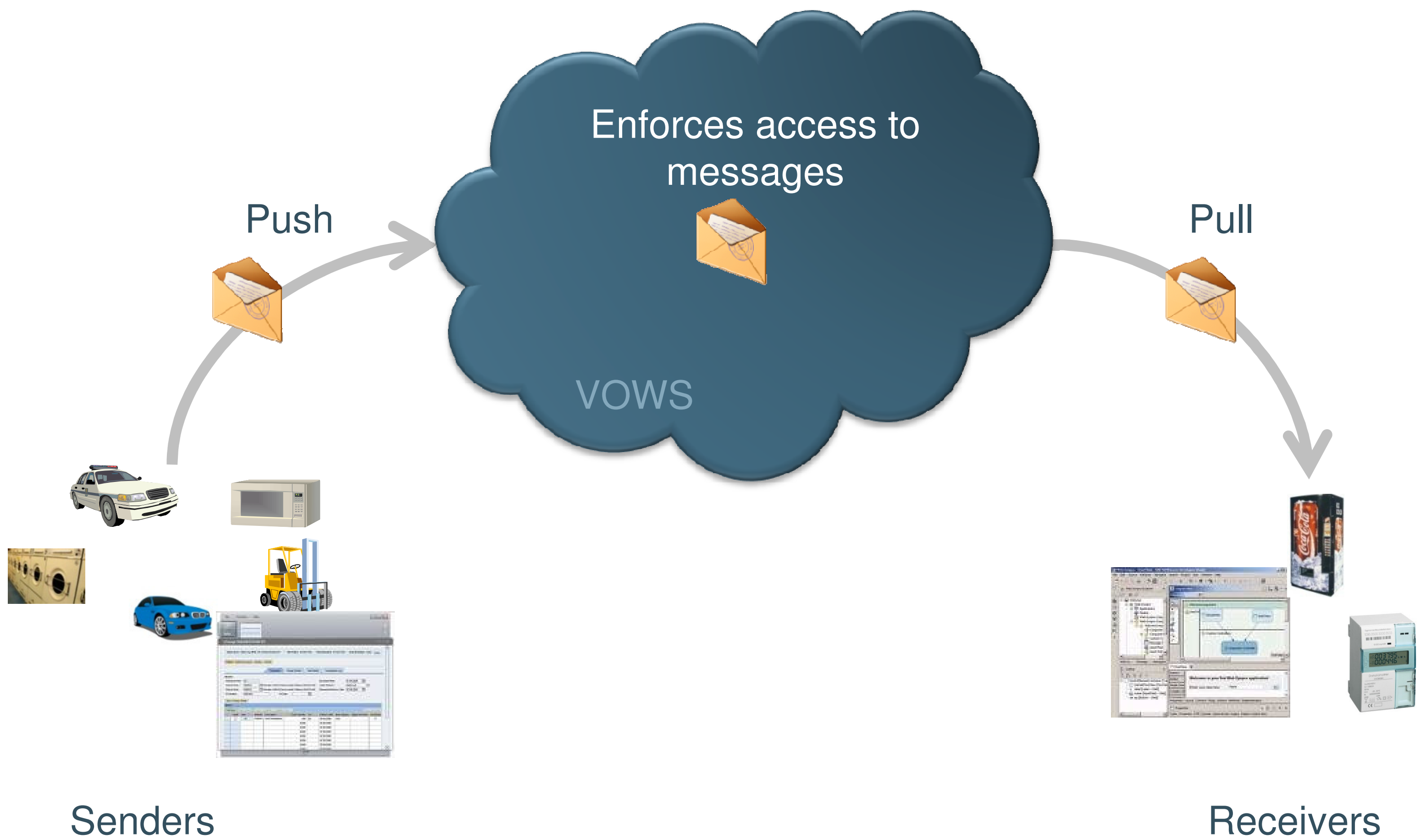


Sumeet Bajaj

Tino Breddin

Martin Stein

The Virtual Object Warehousing Service



Research Goal

Functional requirements

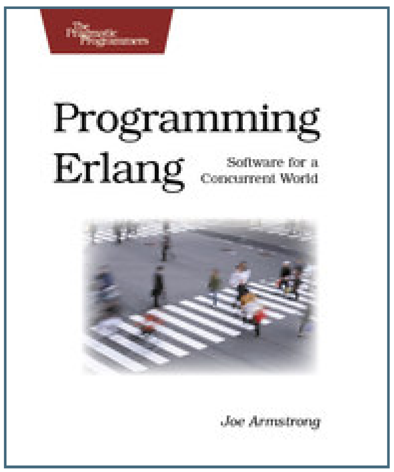
- Connect Devices to enterprise applications
- Asynchronous communication
- Enable communication across enterprise boundaries
- Information is time bound

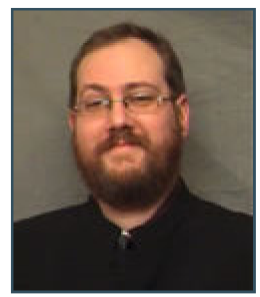
Non-Functional requirements

- Scalability
- Availability
- Performance

Erlang?

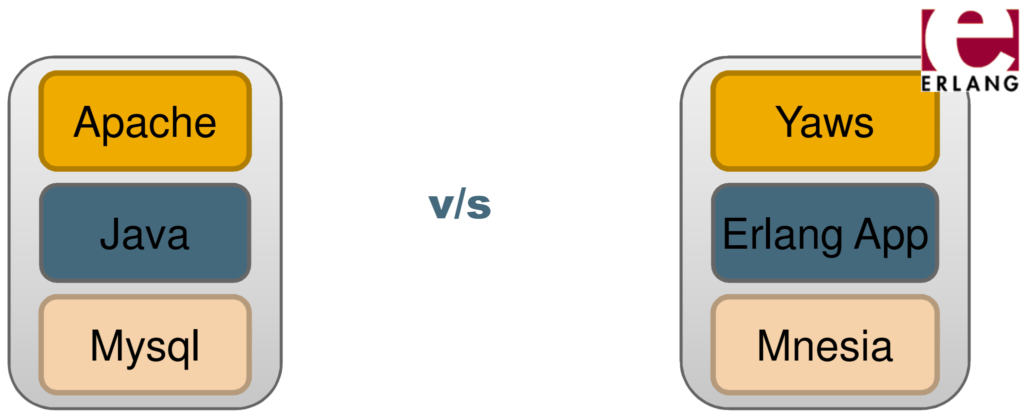
- First source – just a tip
- First reaction : *“Never heard of it“*





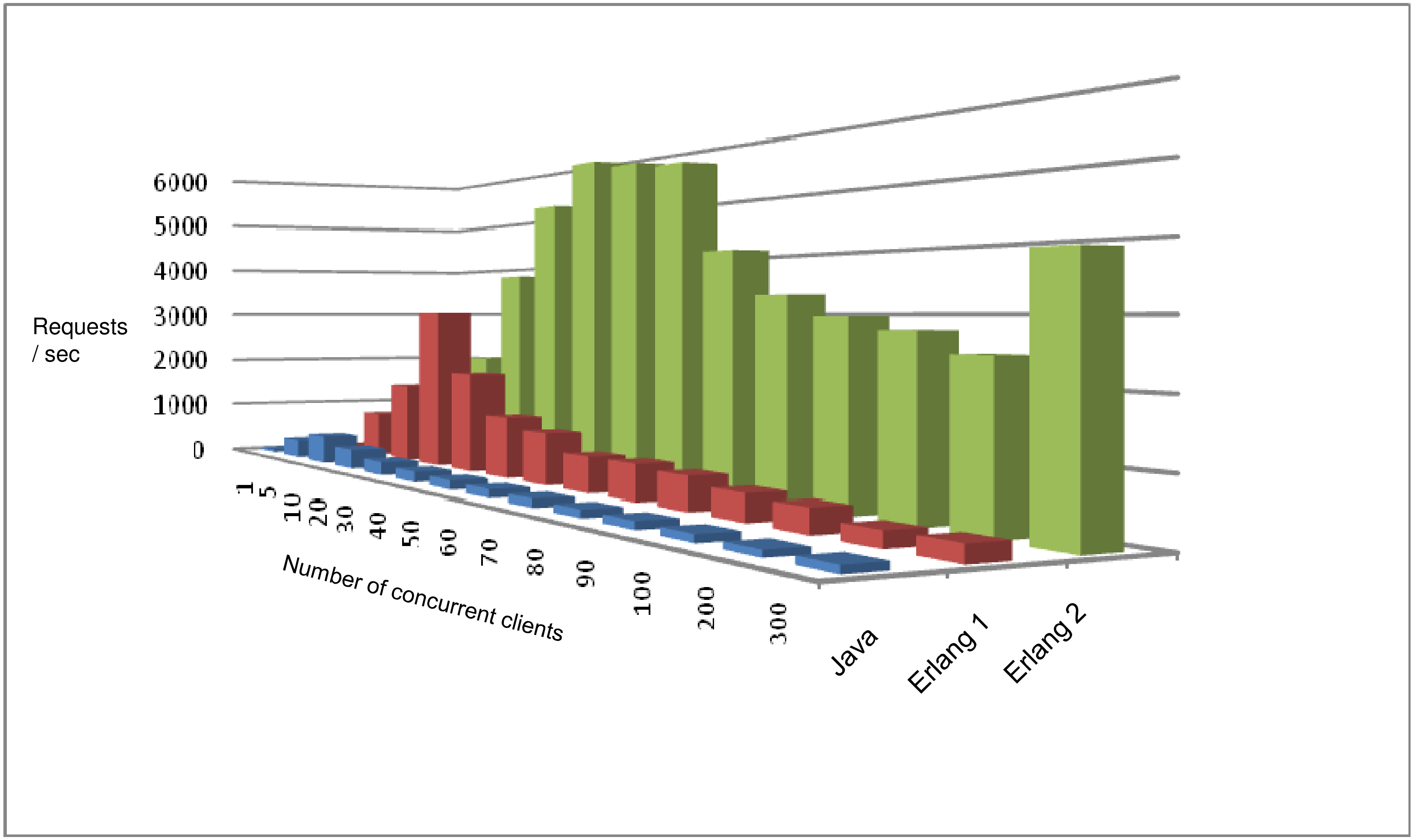
Stage 1 - Learn

Learn by implementation



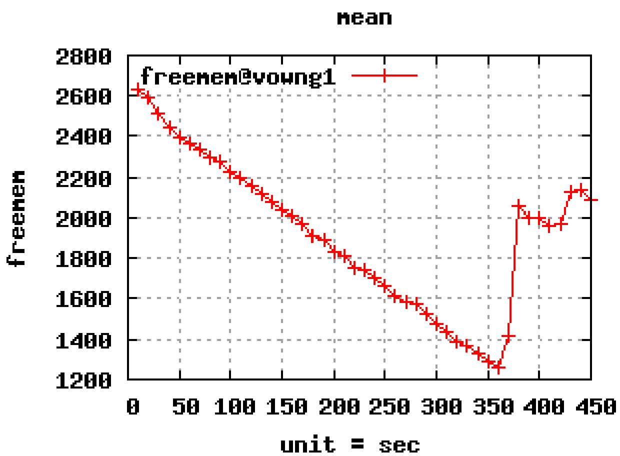
- Learn with hands-on experience
- Use as a subject for performance comparisons
- Evaluate Erlang

Initial Results – Look Promising



Erlang 1 : Disk based storage in mnesia
Erlang 2 : Main memory based storage in mnesia

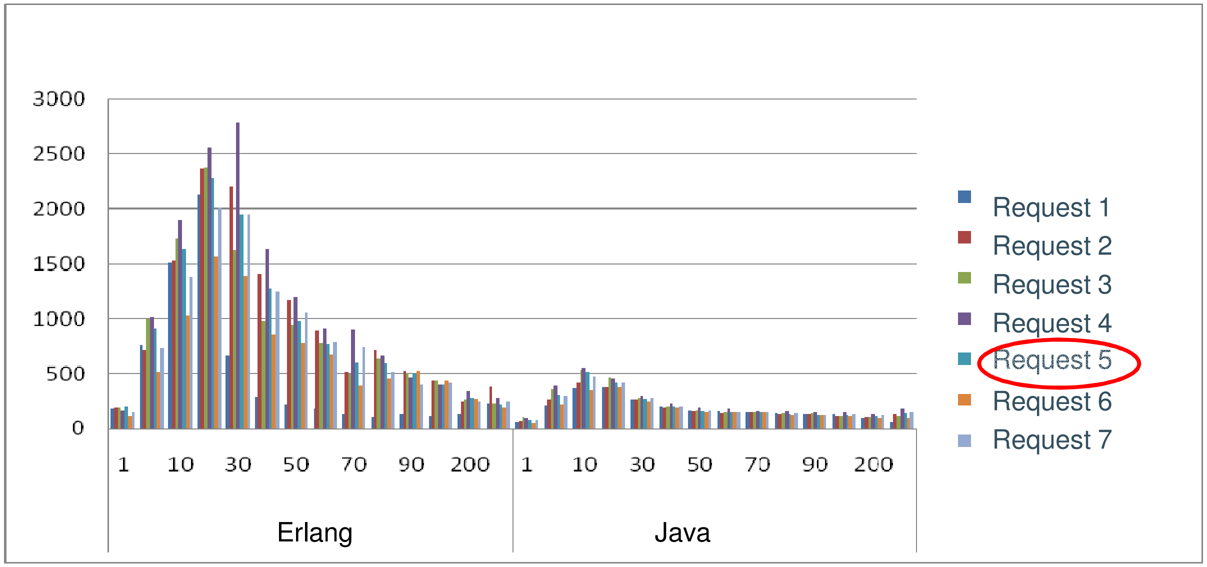
Thorough testing reveals otherwise



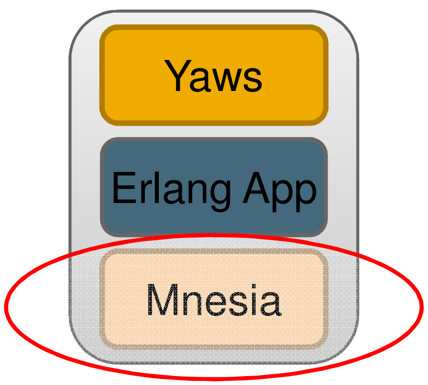
Erlang implementation outperformed Java implementation with message size up to to 1 kb

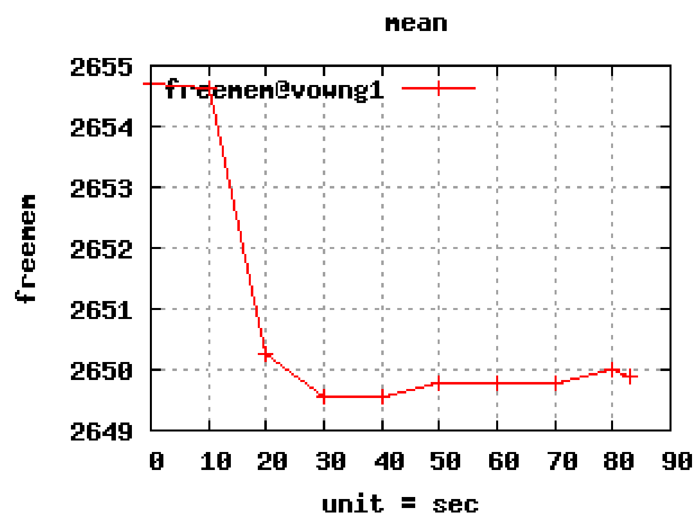
BUT, Java implementation was up to 2-3 times faster with message size set to >= 100k

Investigation reveals the problem



Guilty operation was a qlc





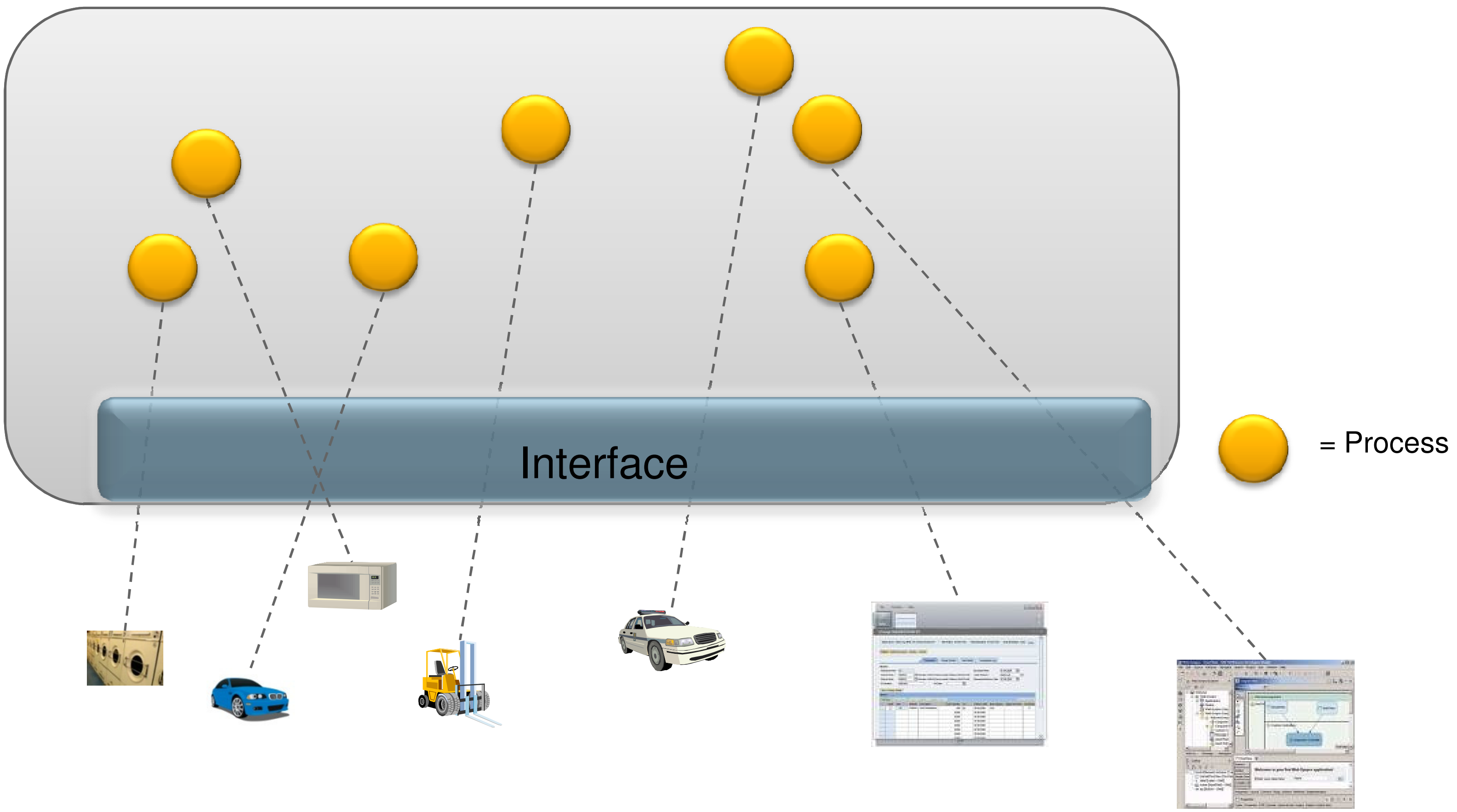
What does this mean?

- Nothing wrong with the tools, but with how they are being used
- The most important factor is the system architecture
- Say ‘YES’ to Erlang

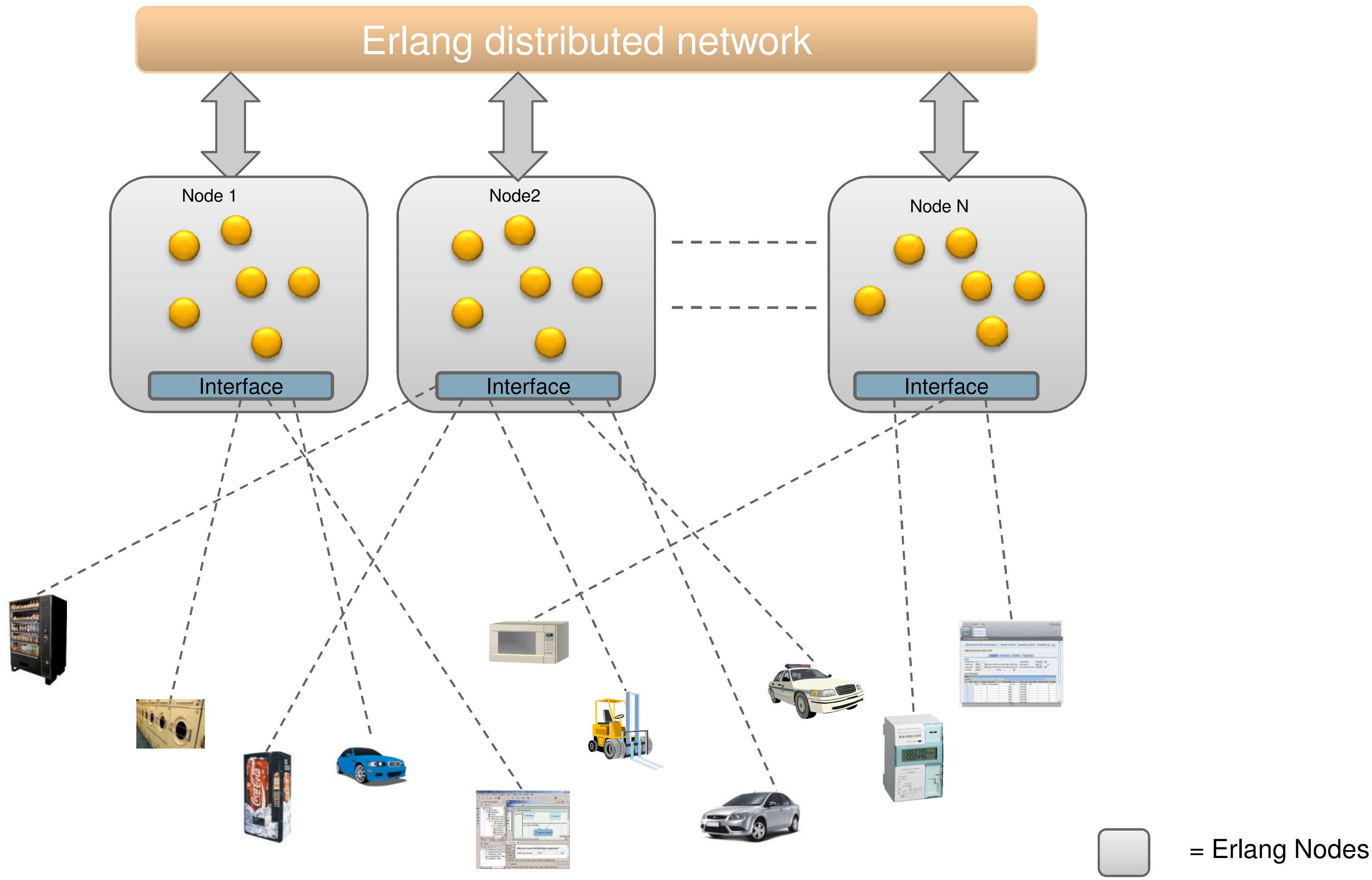
Stage 2 - Rethink

Experiment

Erlang Processes + Distribution + Supervision Trees

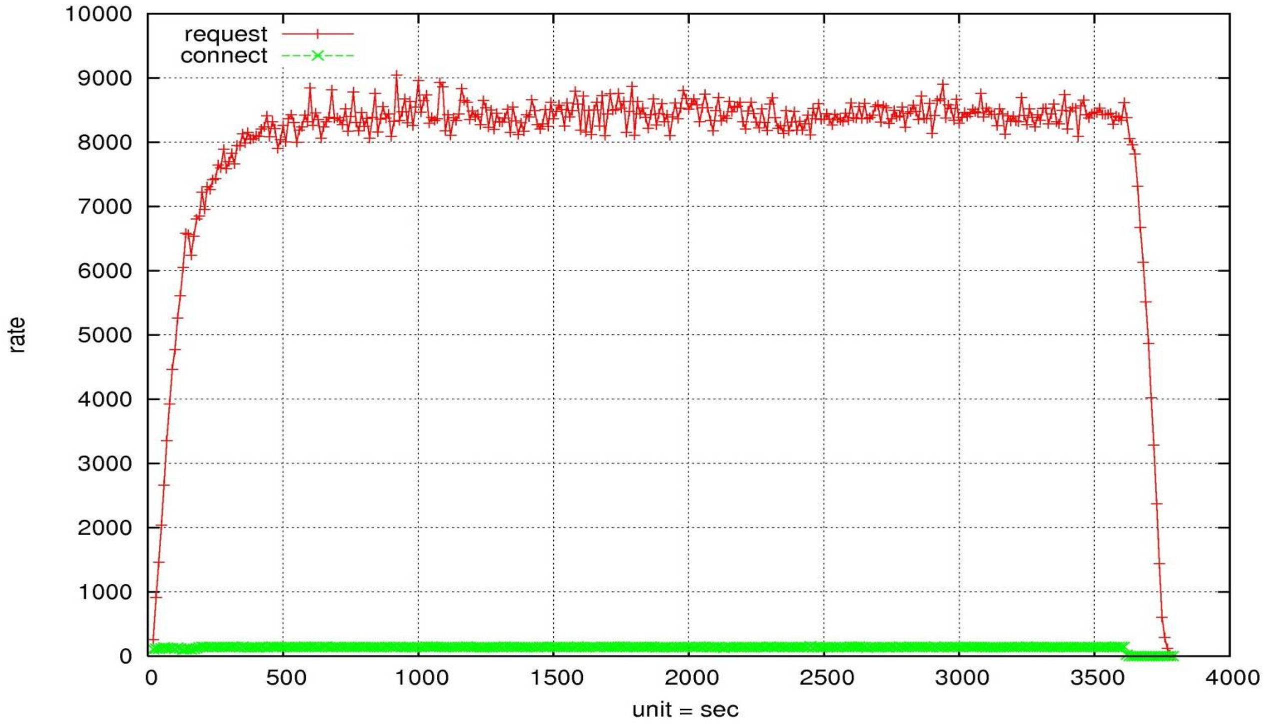


Experiment



Erlang distributed network
Node 1
Node2
Node N
Interface
Interface
Interface
= Erlang Nodes
= Process

Results



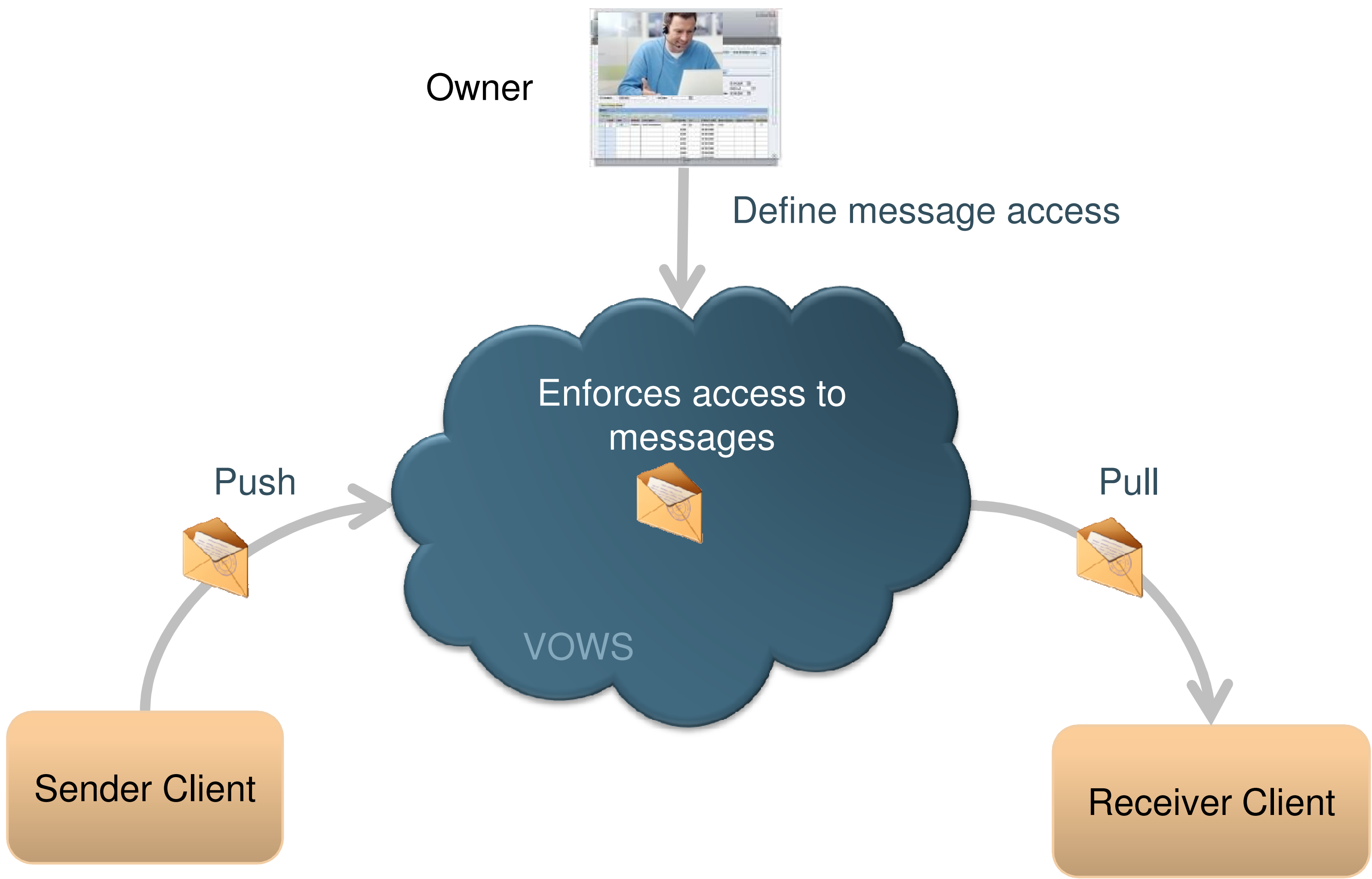
8 servers
~ 80,000 clients

Close to linear scalability up to 8 servers

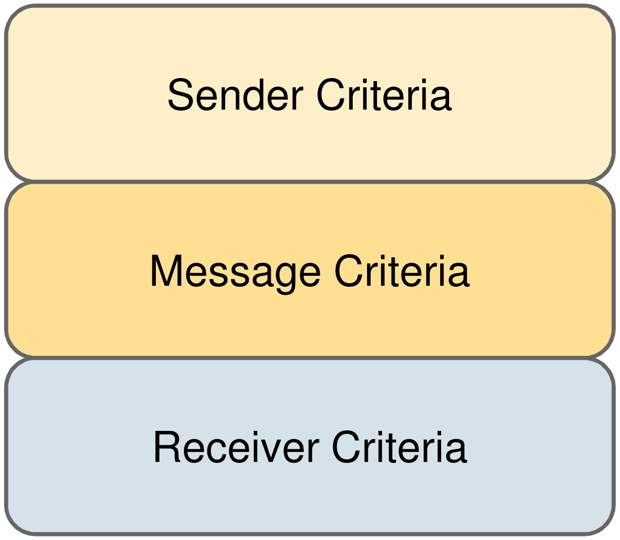
Sample Application: Car Tracking

Stage 3 – Got it!

Virtual Object Warehousing Service



Matching Rule



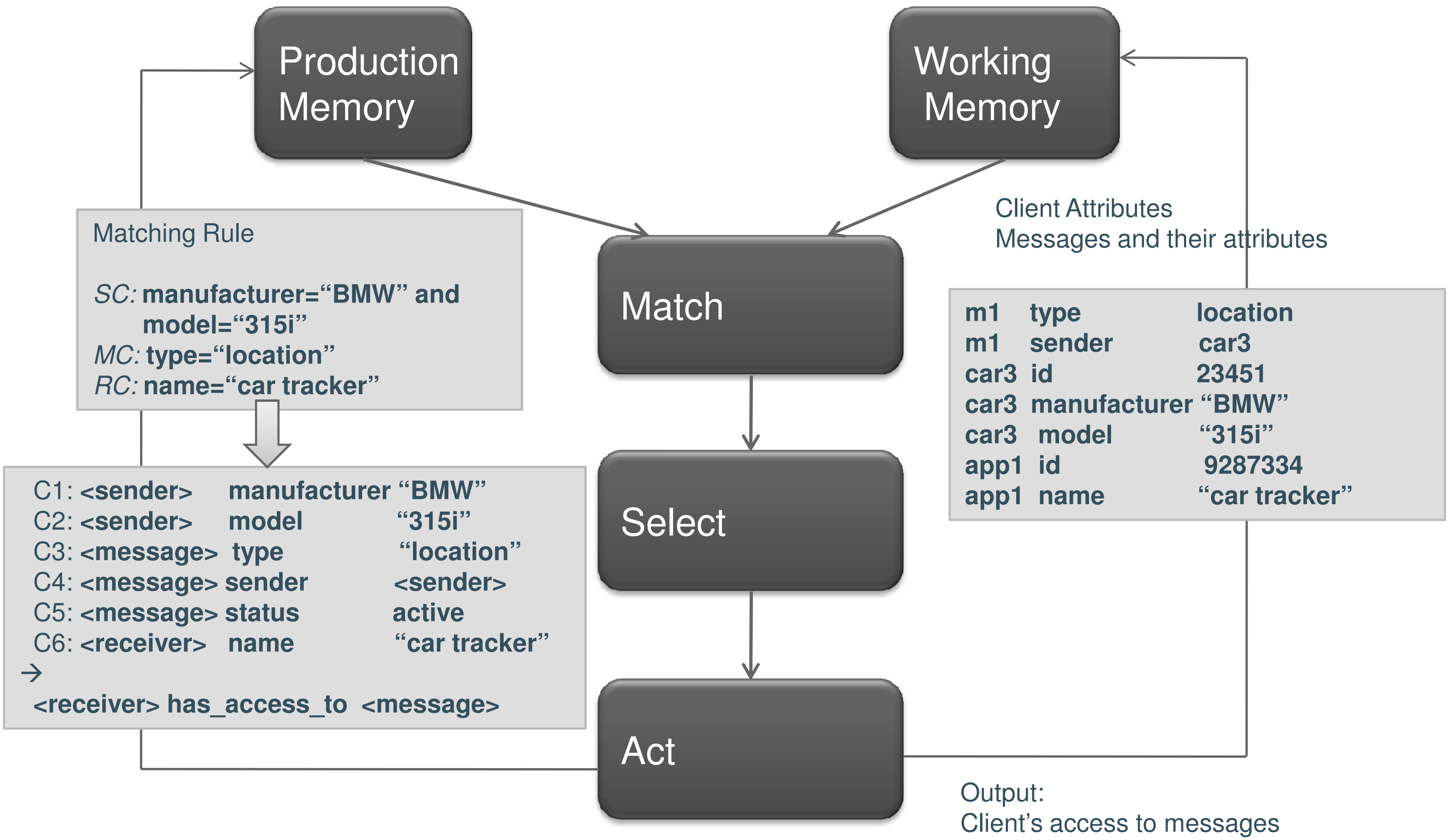
- Applies to client attributes
- e.g. status="on duty", color="red"

- Applies to message attributes
- e.g. type ="request"

- Applies to client attributes
- e.g. model="7H1"

If	the sender meets the Sender Criteria
and	the sender sends the message
and	the message meets the Message Criteria
and	the message is not expired
and	the receiver meets the Receiver Criteria
then	permit receiver to pull the message

Production System



Key design considerations

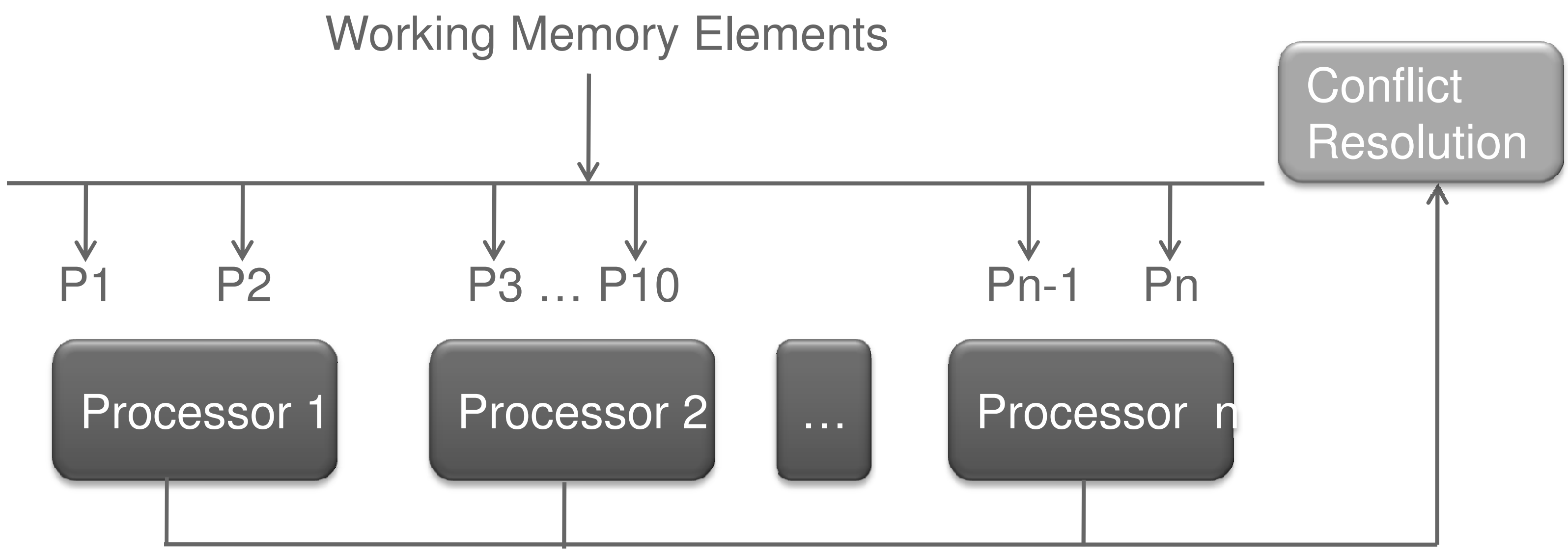
- Very large working memory with very frequent changes
- Frequent changes to production rules (additions/updates/deletes)
- Ability to fire multiple productions in parallel

Sequential

- Was disregarded early on in the research (not suitable for high performance requirements)
- The cross-product effect can only be reduced by parallel processing techniques
- Hard to change production memory

Production Level Parallelism

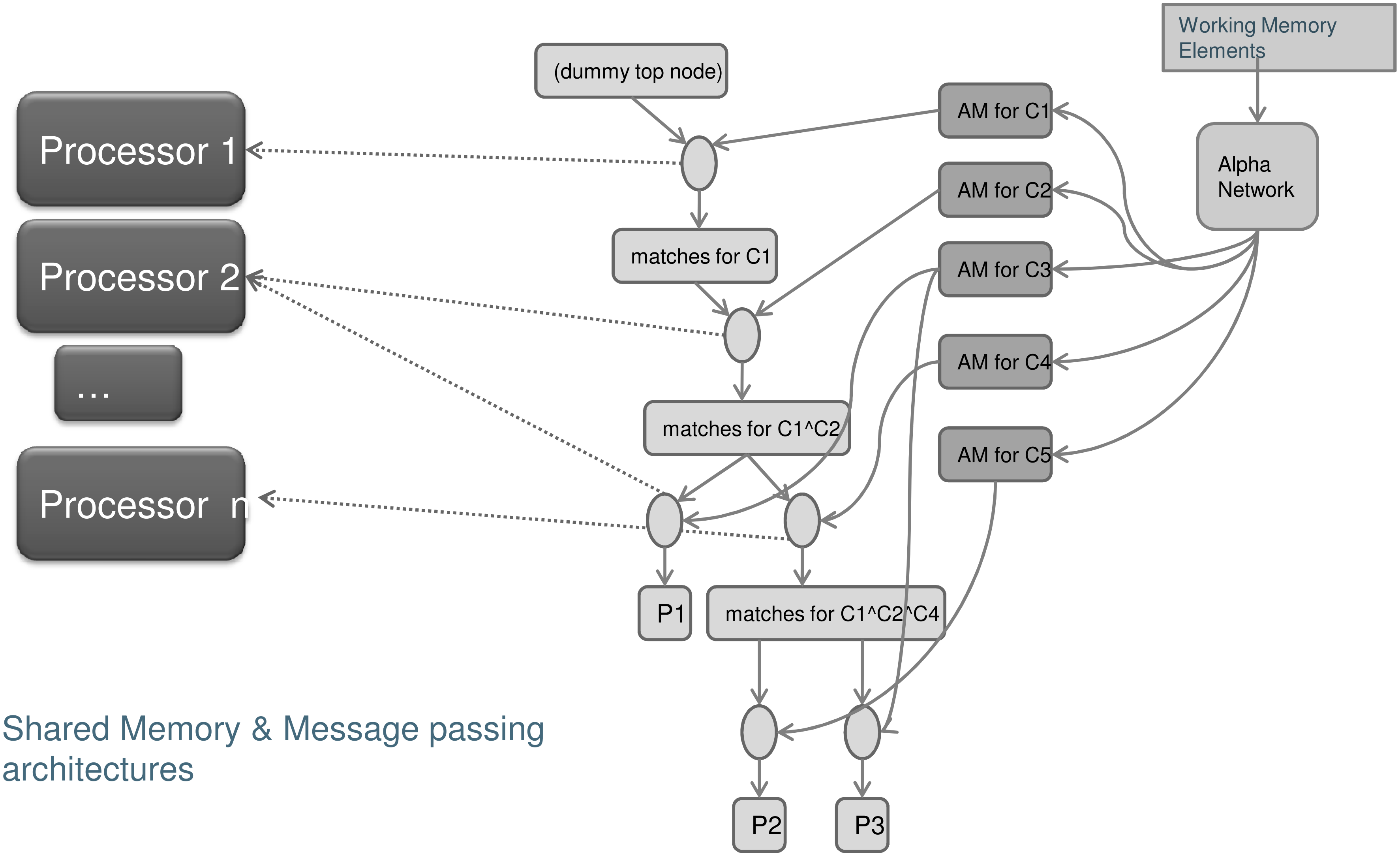
- Assign entire productions to processors



- No communication required between processors for matching
- No synchronization overhead in match phase
- Sharing of computations is limited. Conflict resolution can still be a bottleneck
- Large variations in processing requirements of productions
- Each Rete network is still evaluated sequentially

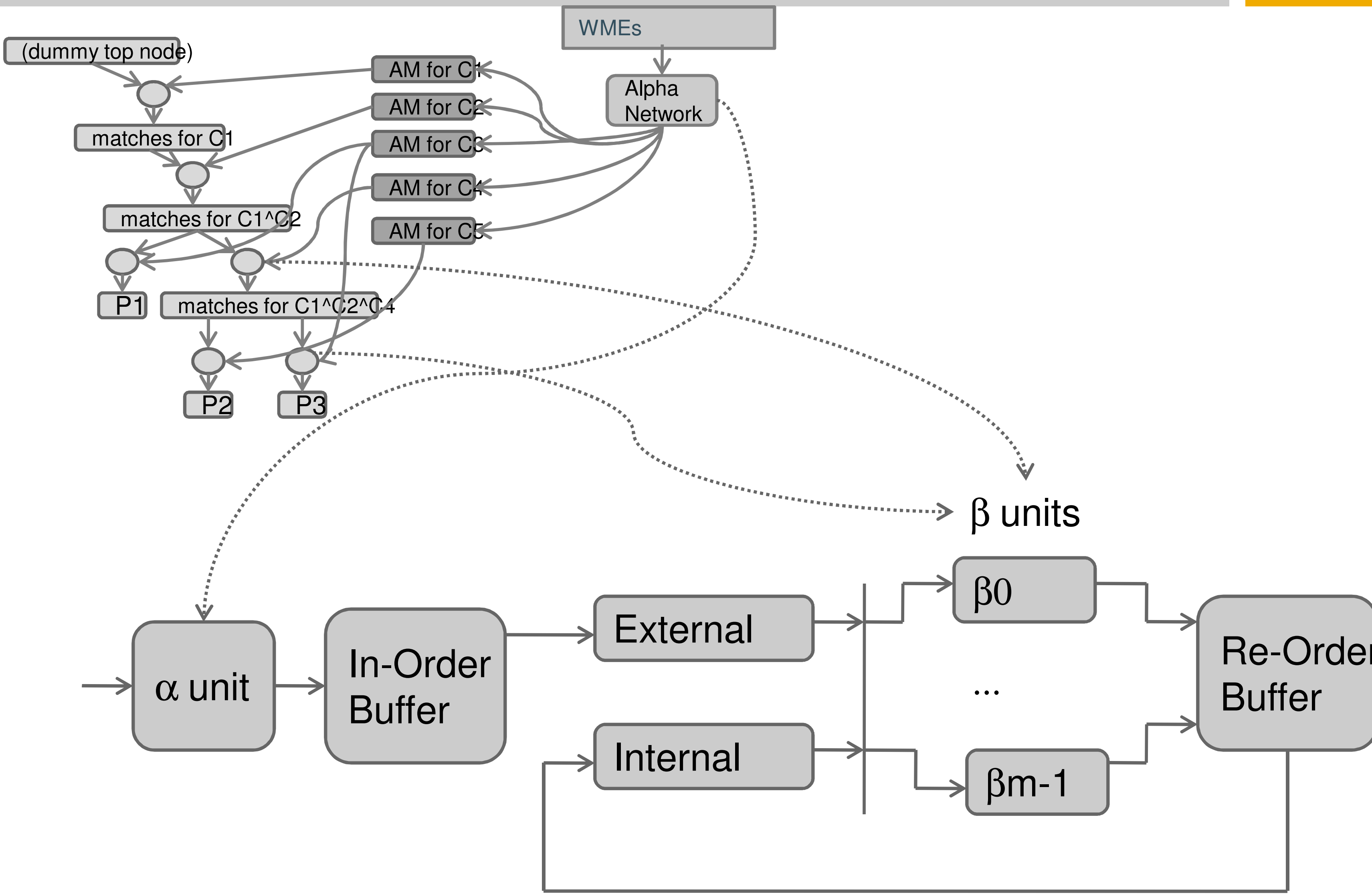
Node Level Parallelism

- Assign nodes to processors



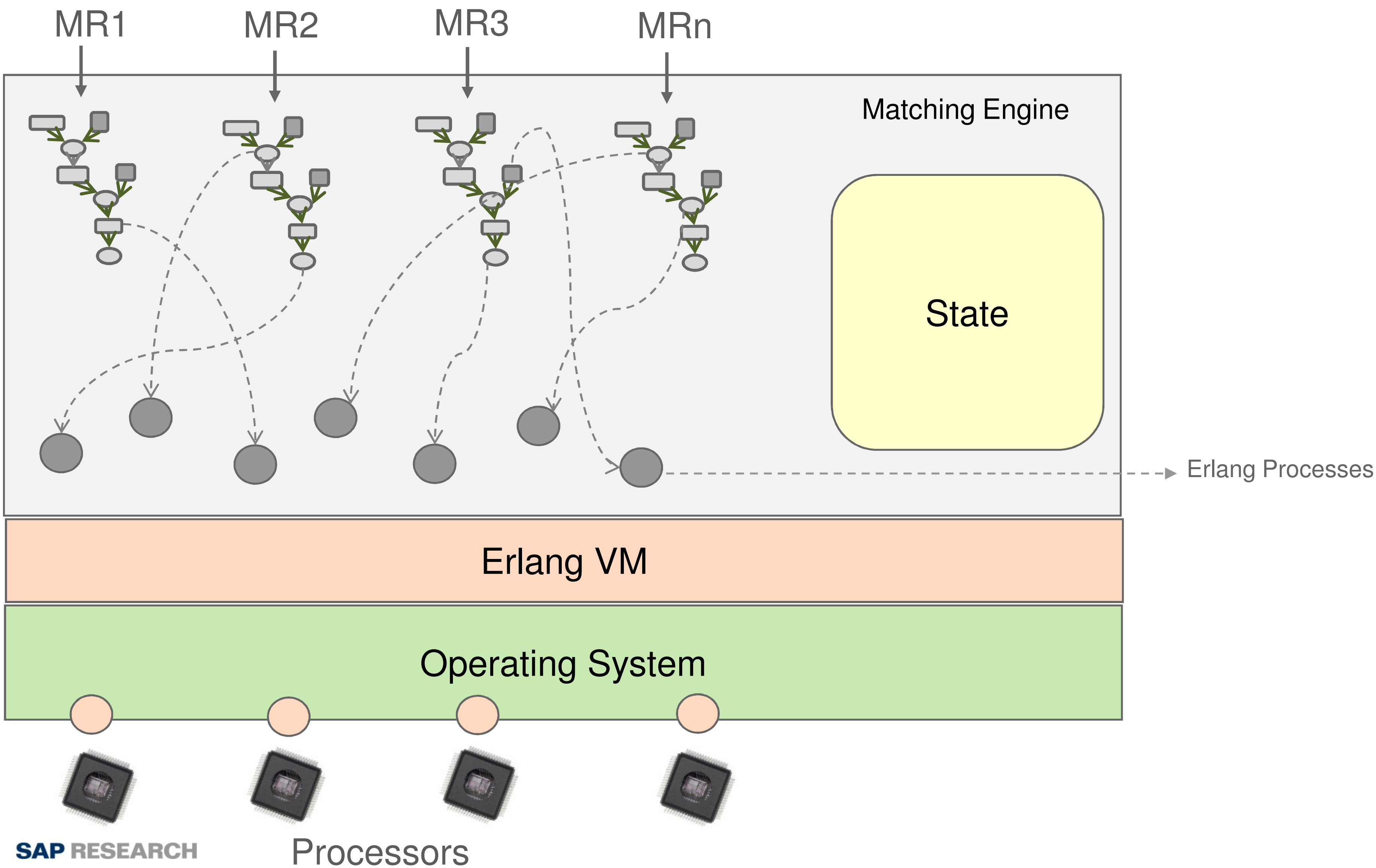
- Shared Memory & Message passing architectures

Parallel Production System Architecture Level 2



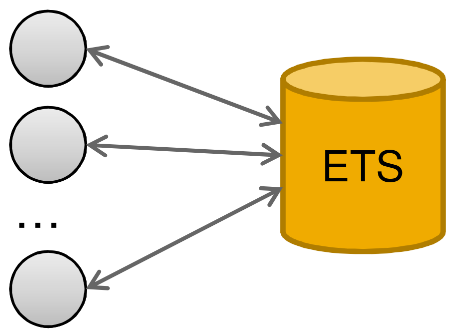
- Thesis: “A Parallel Architecture for Serializable Production Systems” – Jose Nelson Amaral ‘94

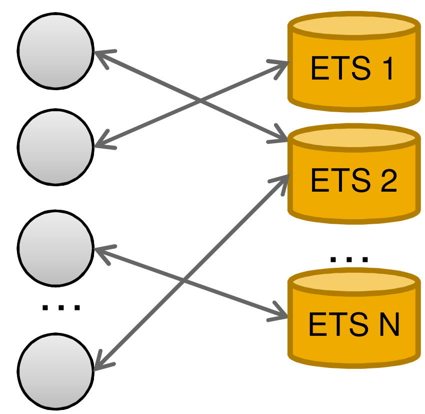
Matching Engine



Use Shared Memory or not?

- ETS tables as shared memory



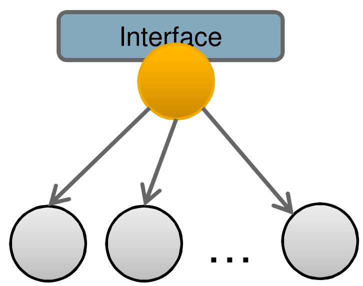


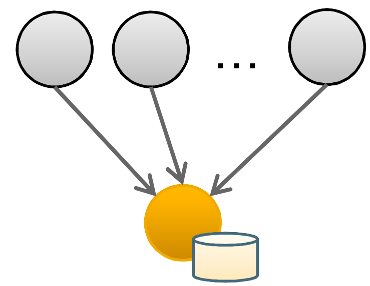
- Access to ETS becomes a bottleneck

- Addition hash operation before insert / lookup. H(Key) => which ETS table to use
- Limited to key based lookups
- Record level locks?

Lessons

- Special processes





Shared resource

- Test & improve

- Application specific
  - Improve by Iterations: 100/s -> 26K/s

Personal Thoughts

- Syntax is cryptic / archaic? Who says?
- Easy to learn
- Well suited to exploit multi-core. Exploit shared memory architectures as well
- Code sizes are remarkably small in comparison
- Do something about strings!

Thank you!