RE^{X}: A Development Platform and Online Learning Approach for Runtime emergent software systems

Barry Porter, Matthew Grieves, Roberto Rodrigues Filho and David Leslie

School of Computing and Communications

Department of Mathematics and Statistics

Lancaster University, UK

b.f.porter@lancaster.ac.uk

MOTIVATION

- Modern software remains **highly complex** to design, implement, maintain & configure, particularly for dynamic environments
  - this causes high development costs and under-performing code

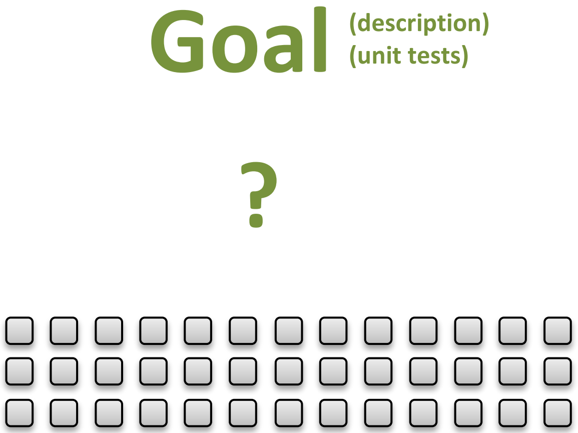
- The state of the art in solving this is **self-adaptive systems**, which exhibit some awareness of *self* and of *environment*
  - However, these approaches relate only to the configuration element of systems development, and are also very manual in their definition, by:
    - Designing the base system as a non-adaptive one
    - Deciding which points of that system should be adaptable
    - Writing rules to determine how and when adaptation happens

CONCEPT OVERVIEW

- We propose a paradigm of **continuous self-assembly**, in which the *initial construction* of software and its *later adaptation* are one continuous machine-driven process

Concept overview

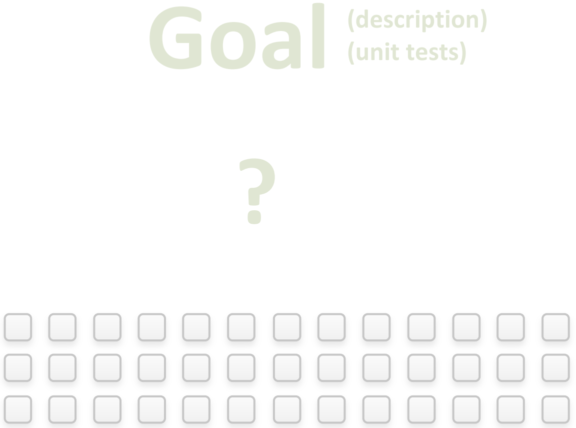
- We propose a paradigm of **continuous self-assembly**, in which the *initial construction* of software and its *later adaptation* are one continuous machine-driven process



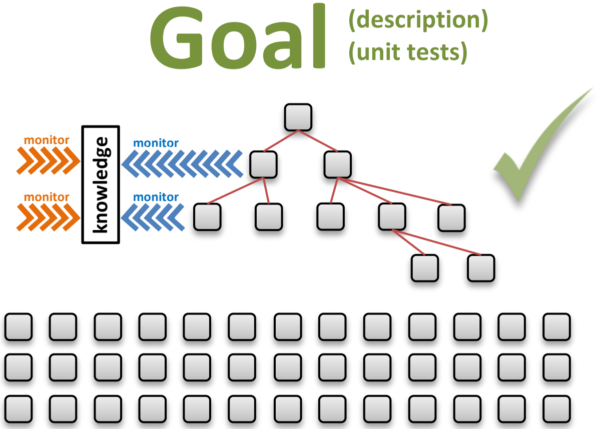
Start with a goal and a pool of behavior fragments

CONCEPT OVERVIEW

- We propose a paradigm of **continuous self-assembly**, in which the *initial construction* of software and its *later adaptation* are one continuous machine-driven process



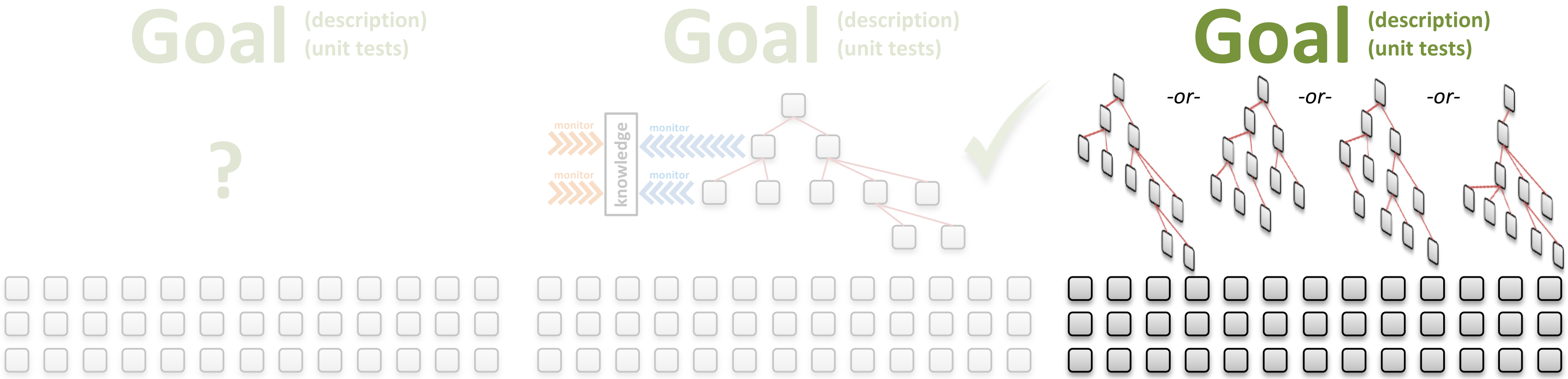
Start with a goal and a pool of behavior fragments



*Form a working system for the goal, deploy in production environment and monitor **performance** and **environment***

Concept overview

- We propose a paradigm of **continuous self-assembly**, in which the *initial construction* of software and its *later adaptation* are one continuous machine-driven process



Start with a goal and a pool of behavior fragments

*Form a working system for the goal, deploy in production environment and monitor **performance** and **environment***

*Learn & find (or create) variants to better **shape the system to its environment***

CONTRIBUTIONS

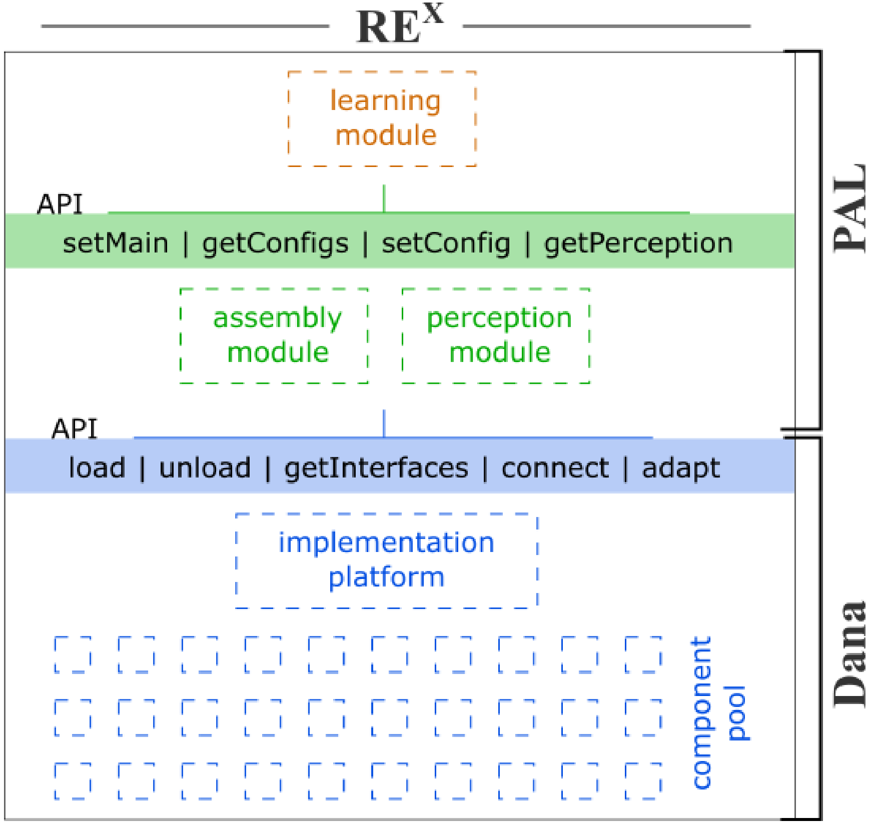
- **Implementation platform (*Dana*):** A programming language with which to create small software building blocks that can be assembled into emergent systems, with near-zero-cost runtime adaptation for online exploration.

- **Perception, assembly and learning framework (*PAL*):** A framework built with Dana to discover & assemble emergent software, perceive its effectiveness and deployment conditions, and feed perception data to online learning.

- **Online learning approach:** An application of statistical linear bandits, using Thompson sampling, to help solve the search space explosion inherent in our approach, by sharing beliefs about components across possible configurations.

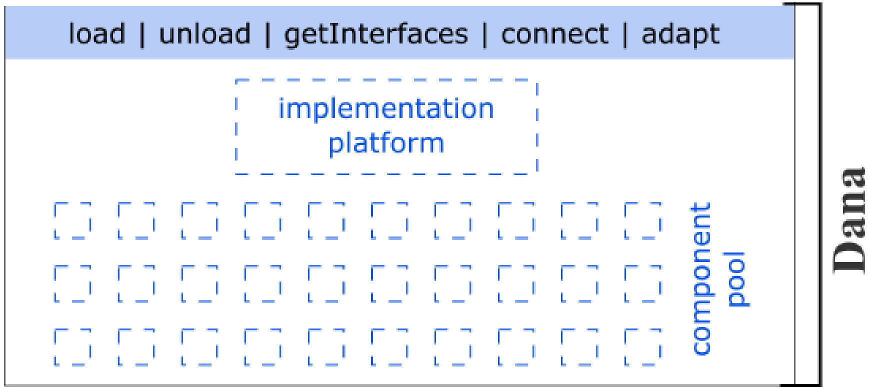
Example system: an emergent, self-assembling web server

APPROACH IN DETAIL



APPROACH IN DETAIL – IMPLEMENTATION PLATFORM

- Uses a **component-based development** paradigm, but:
  (i) infused in a generalised systems programming language;
  (ii) supporting very fast, fine-grained runtime adaptation; and
  (iii) removing the need for wiring diagrams / configurations

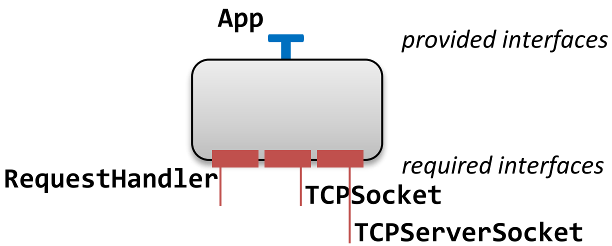


Approach in Detail – Implementation platform

- Uses a **component-based development** paradigm

```
interface RequestHandler {
 void handleRequest(TCPSocket s)
}
```

```
component provides App requires net.TCPSocket,
                          net.TCPServerSocket,
                    request.RequestHandler rh {

 int App:main(AppParam params[]) {
  TCPServerSocket host = new TCPServerSocket()
  host.bind(TCPServerSocket.ANY_ADDRESS, 8080)

  while (true) {
   TCPSocket client = new TCPSocket()
   if (client.accept(host))
    rh.handleRequest(client)
  }

  return 0
 }
}
```

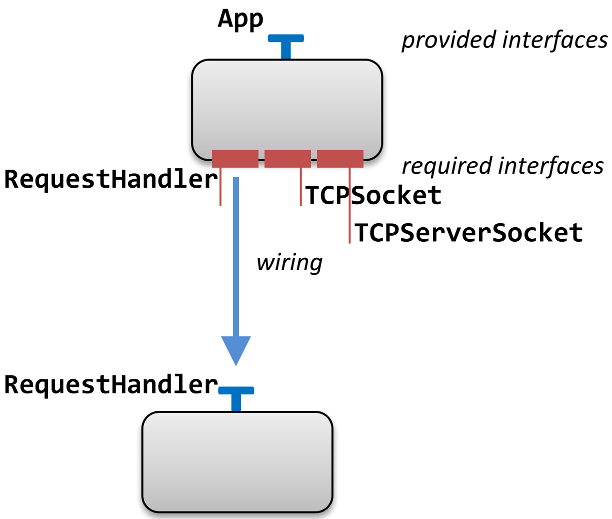


Approach in Detail – Implementation platform

- Uses a **component-based development** paradigm

```
interface RequestHandler {
 void handleRequest(TCPSocket s)
}
```

```
component provides App requires net.TCPSocket,
                          net.TCPServerSocket,
                    request.RequestHandler rh {

 int App:main(AppParam params[]) {
  TCPServerSocket host = new TCPServerSocket()
  host.bind(TCPServerSocket.ANY_ADDRESS, 8080)

  while (true) {
   TCPSocket client = new TCPSocket()
   if (client.accept(host))
    rh.handleRequest(client)
  }

  return 0
 }
}
```

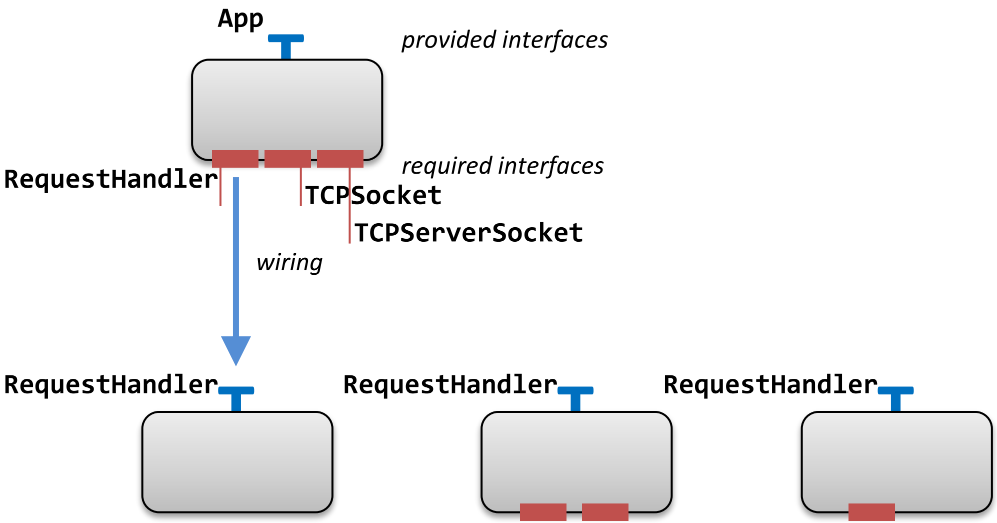


Approach in Detail – Implementation platform

- Uses a **component-based development** paradigm

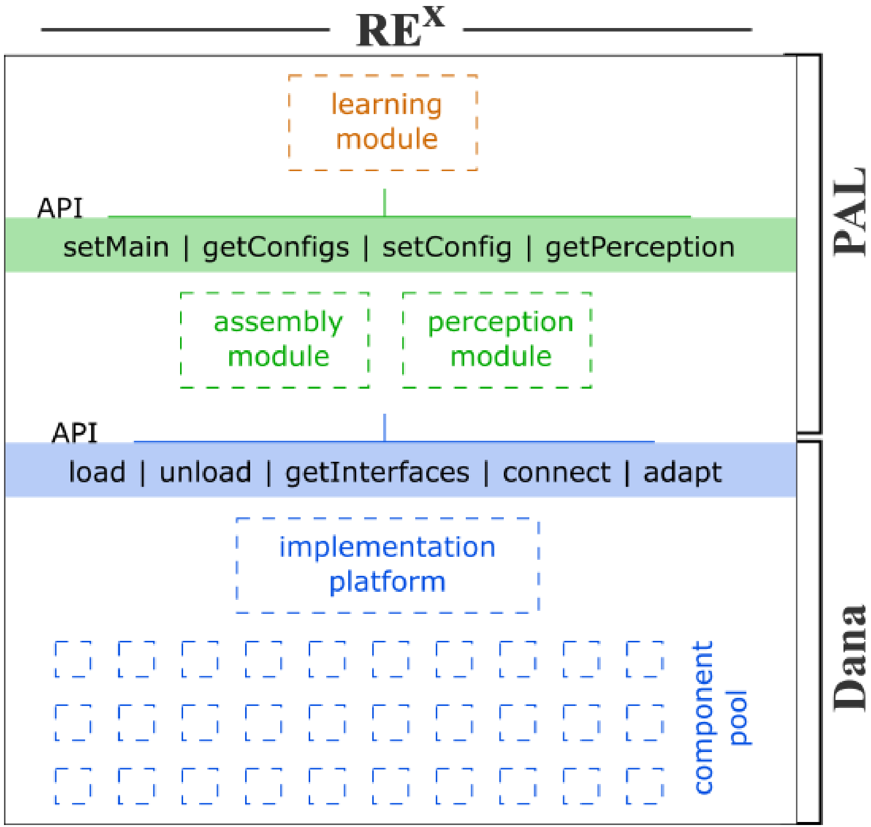
```
interface RequestHandler {
 void handleRequest(TCPSocket s)
}
```

```
component provides App requires net.TCPSocket,
                          net.TCPServerSocket,
                    request.RequestHandler rh {

 int App:main(AppParam params[]) {
  TCPServerSocket host = new TCPServerSocket()
  host.bind(TCPServerSocket.ANY_ADDRESS, 8080)

  while (true) {
   TCPSocket client = new TCPSocket()
   if (client.accept(host))
    rh.handleRequest(client)
  }

  return 0
 }
}
```



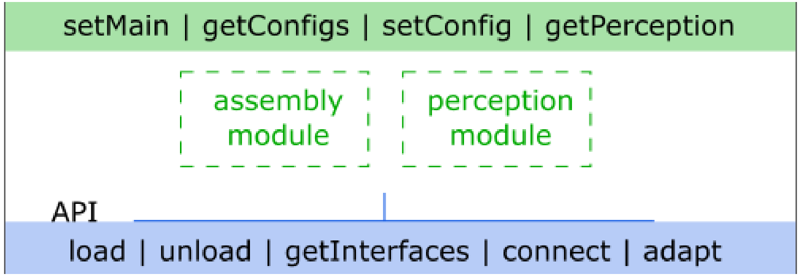
implementation variants: components that provide the same interface but implement it in different ways, yielding different behaviour or performance

APPROACH IN DETAIL



APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



setMain | getConfigs | setConfig | getPerception

assembly module

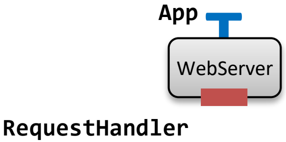
perception module

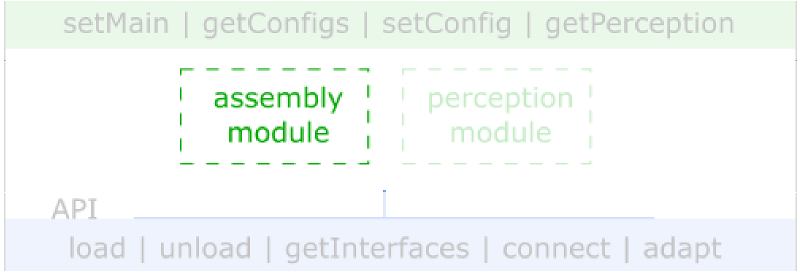
API

load | unload | getInterfaces | connect | adapt

APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

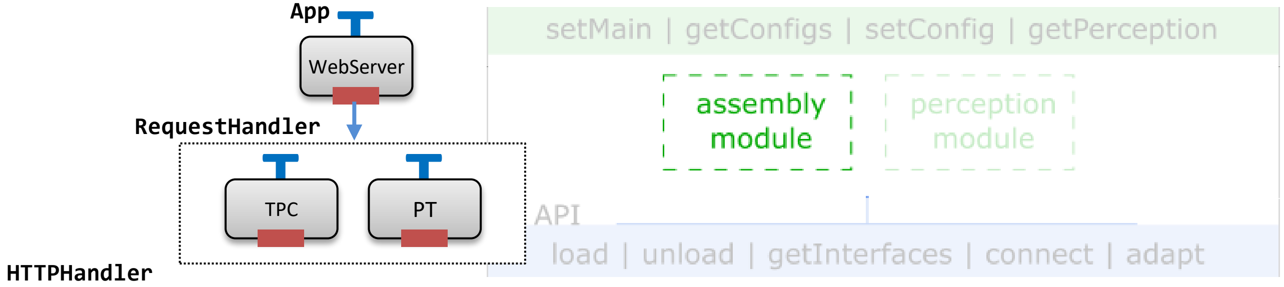
- A way to **abstract entire software systems** for machine learning into: reward; environment; actions




setMain | getConfigs | setConfig | getPerception
assembly module
perception module
API
load | unload | getInterfaces | connect | adapt

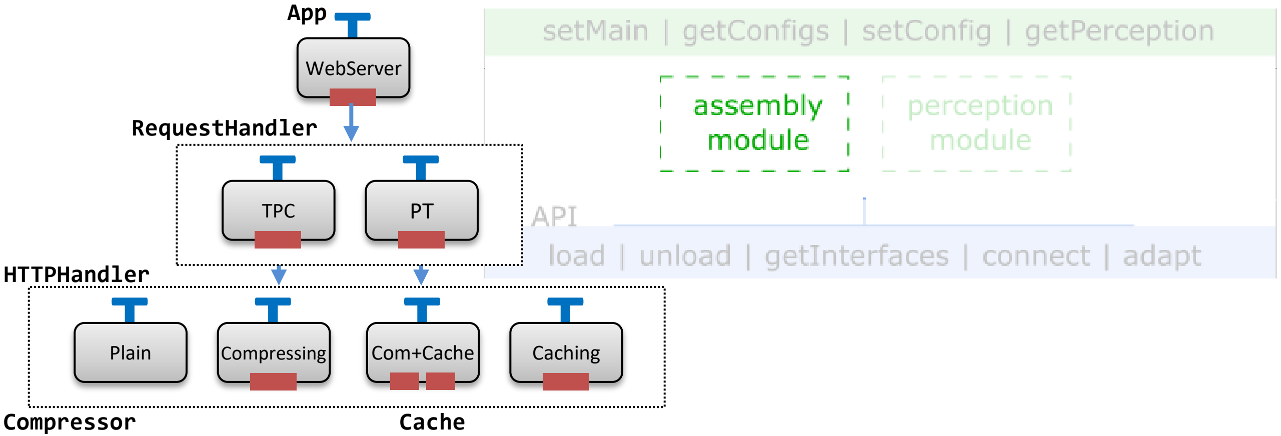
APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions


App
WebServer
RequestHandler
TPC
PT
HTTPHandler
setMain | getConfigs | setConfig | getPerception
assembly module
perception module
API
load | unload | getInterfaces | connect | adapt

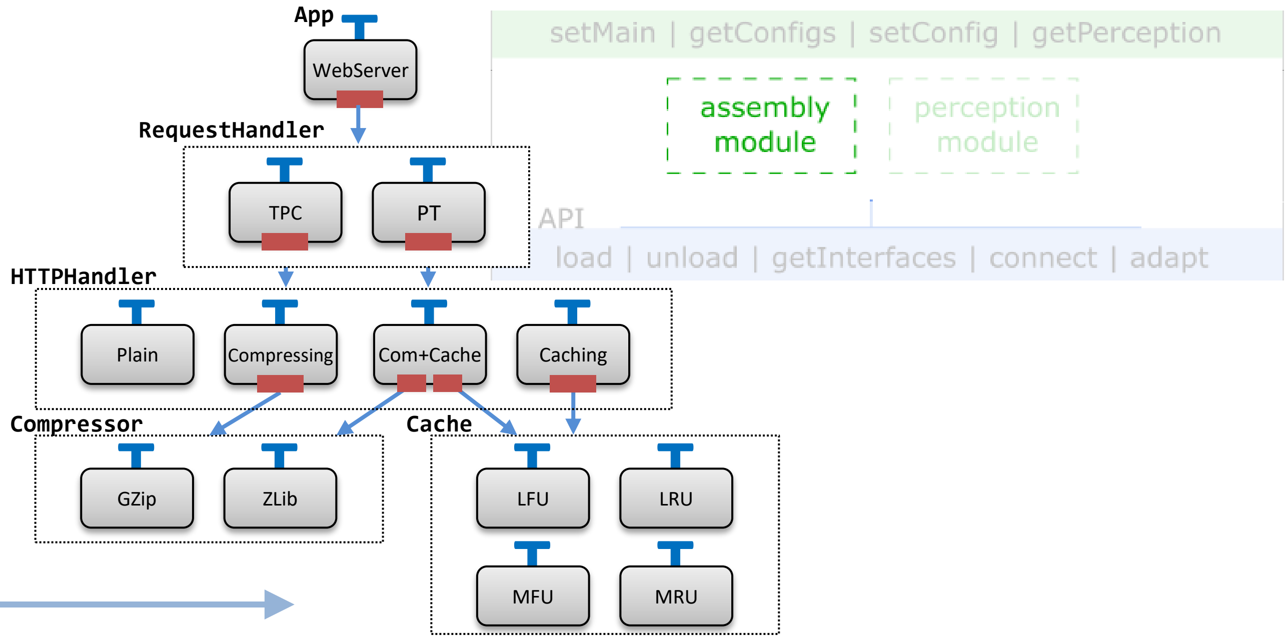
APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



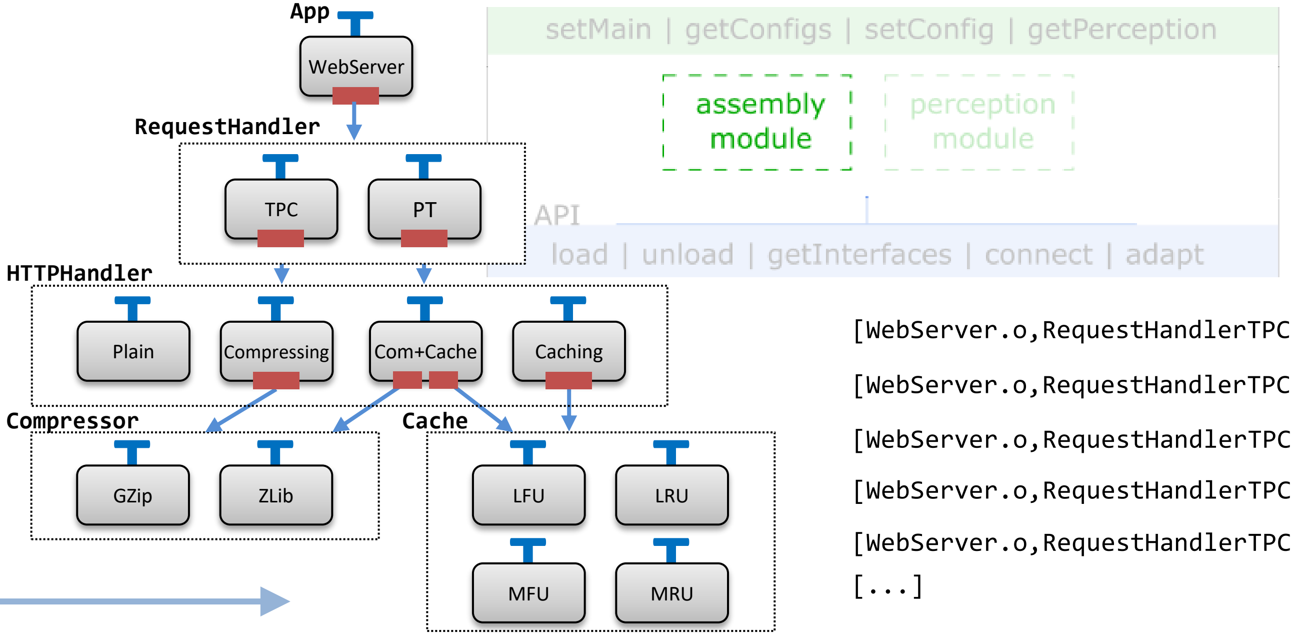
APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

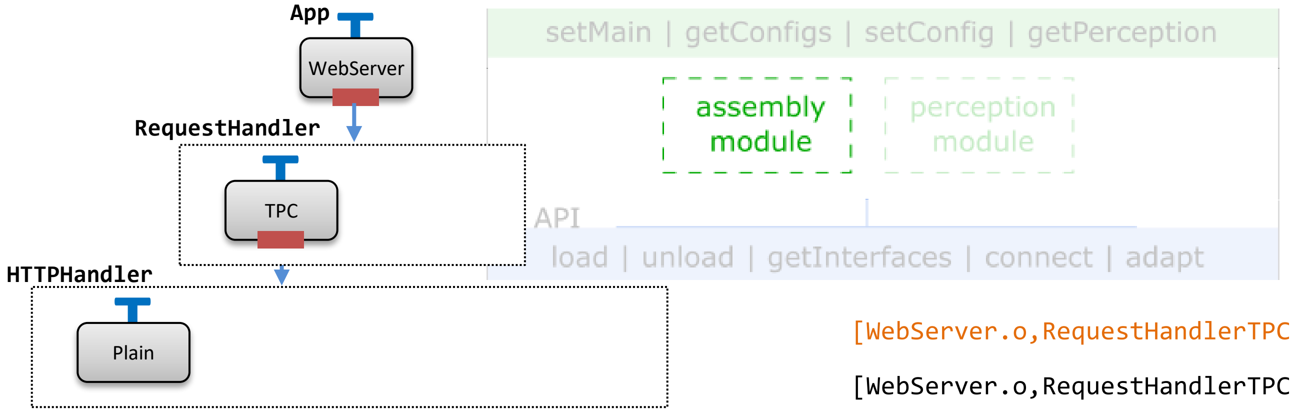
- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



```
[WebServer.o,RequestHandlerTPC.o,HTTPHandler.o,TCP.o,File.o,...]

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerZ.o,GZip.o,TCP.o,...]

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerCZ.o,CacheMFU.o,GZip.o,...

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerCZ.o,CacheLRU.o,GZip.o,...

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerC.o,CacheMFU.o,TCP.o,...]
[...]
```

Approach in Detail – Perception, Assembly and Learning

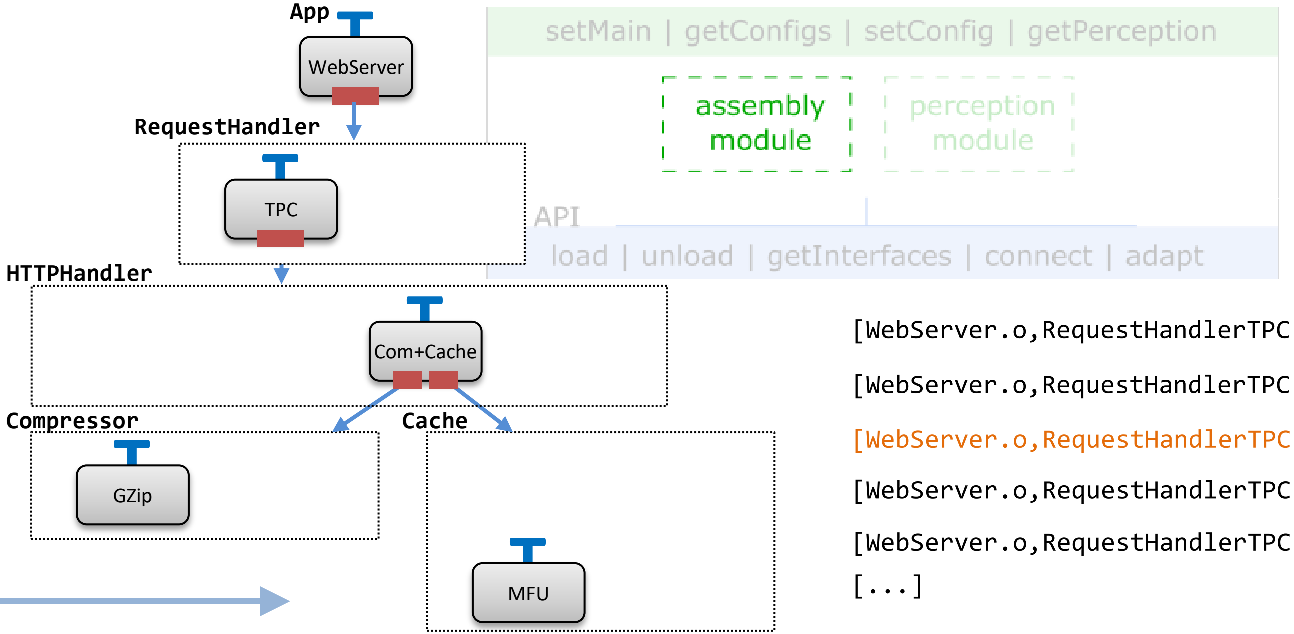
- A way to **abstract entire software systems** for machine learning into: reward; environment; actions


App
WebServer
RequestHandler
TPC
HTTPHandler
Plain
setMain | getConfigs | setConfig | getPerception
assembly module
perception module
API
load | unload | getInterfaces | connect | adapt

```
[WebServer.o,RequestHandlerTPC.o,HTTPHandler.o,TCP.o,File.o,...]
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerZ.o,GZip.o,TCP.o,...]
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerCZ.o,CacheMFU.o,GZip.o,...
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerCZ.o,CacheLRU.o,GZip.o,...
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerC.o,CacheMFU.o,TCP.o,...]
[...]
```

APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

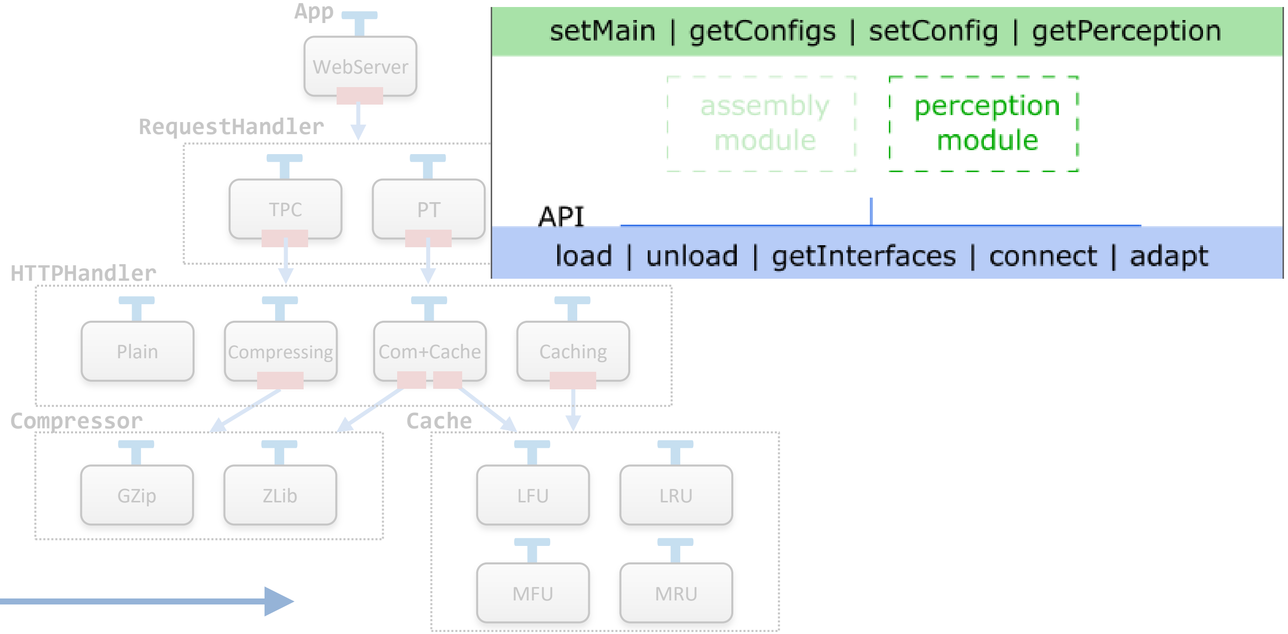
- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



```
[WebServer.o,RequestHandlerTPC.o,HTTPHandler.o,TCP.o,File.o,...]

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerZ.o,GZip.o,TCP.o,...]

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerCZ.o,CacheMFU.o,GZip.o,...

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerCZ.o,CacheLRU.o,GZip.o,...

[WebServer.o,RequestHandlerTPC.o,HTTPHandlerC.o,CacheMFU.o,TCP.o,...]

[...]
```

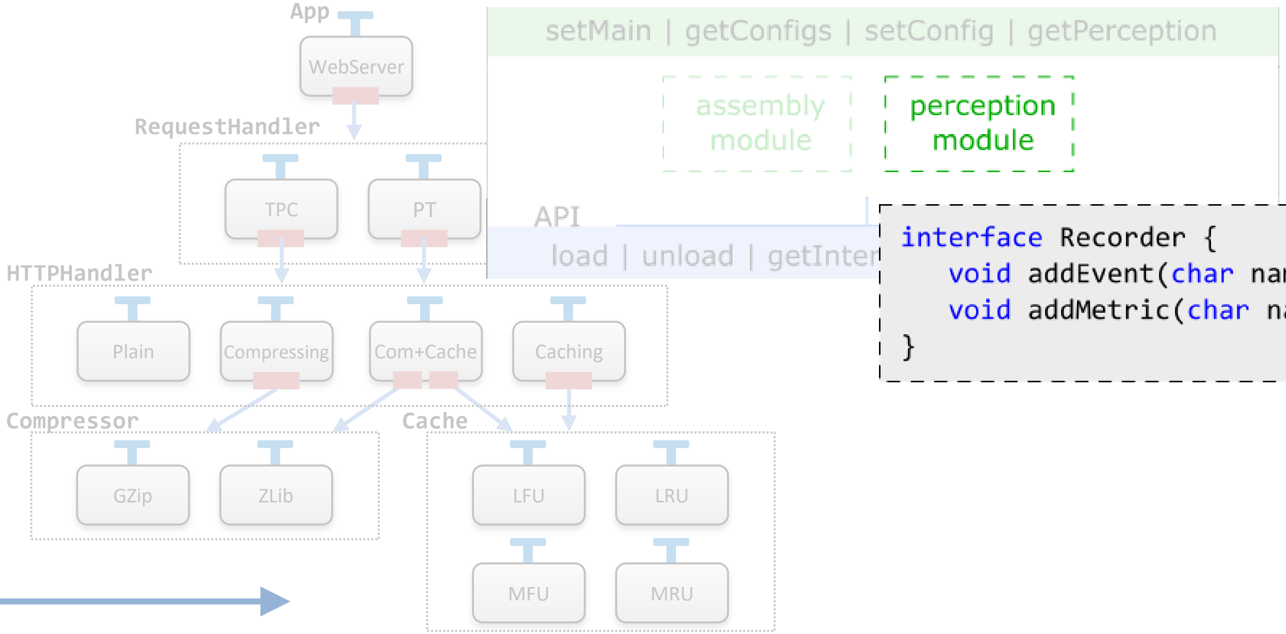
Approach in detail – Perception, Assembly and Learning

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

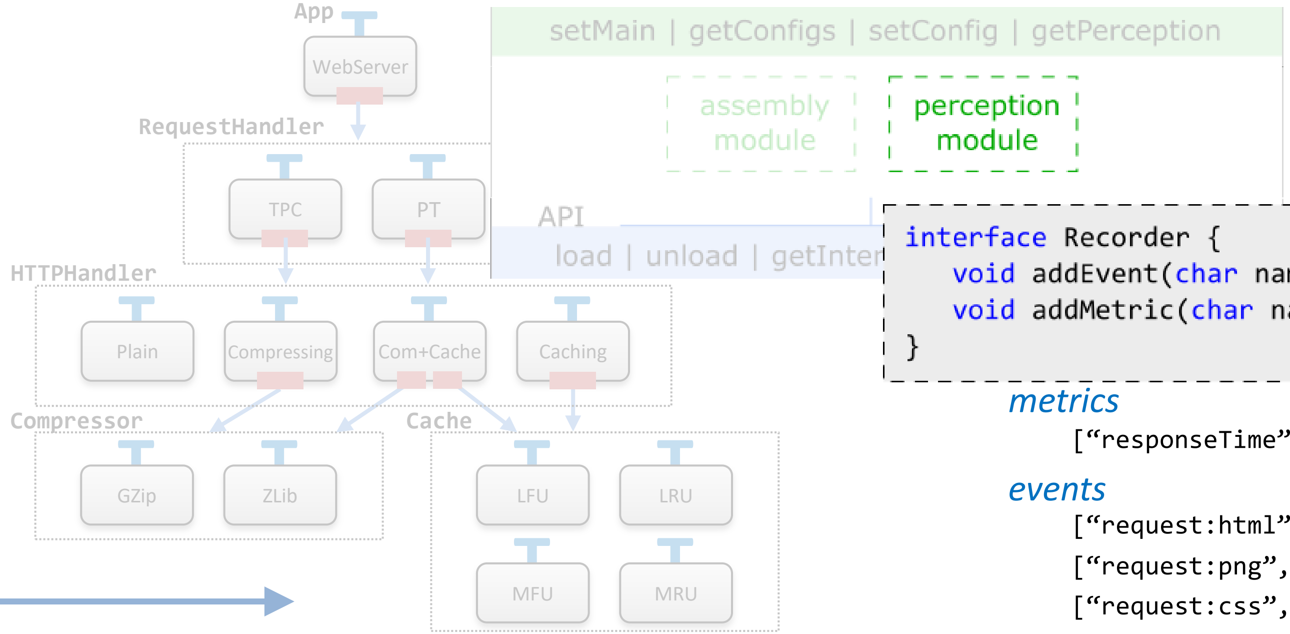
- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



```
interface Recorder {
   void addEvent(char name[], int value)
   void addMetric(char name[], int value, bool preferHigh)
}
```

APPROACH IN DETAIL – PERCEPTION, ASSEMBLY AND LEARNING

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions



```
interface Recorder {
   void addEvent(char name[], int value)
   void addMetric(char name[], int value, bool preferHigh)
}
```

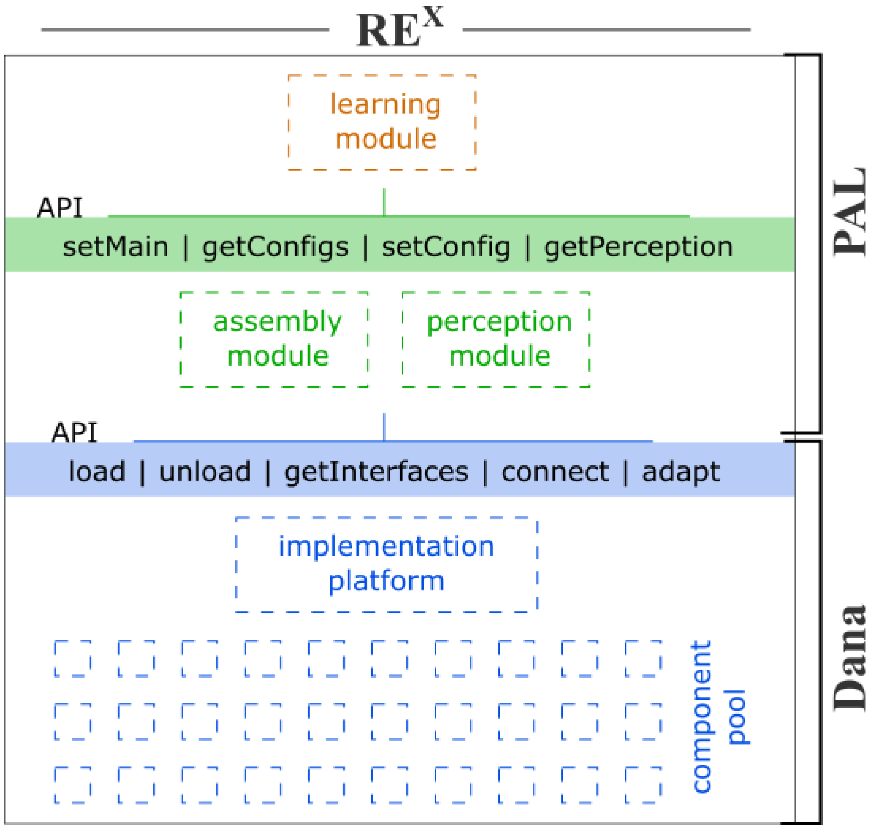
metrics

```
["responseTime", 49, 32, true]
```

events

```
["request:html", 5392, 14]
["request:png", 16702, 12]
["request:css", 2420, 6]
```

APPROACH IN DETAIL



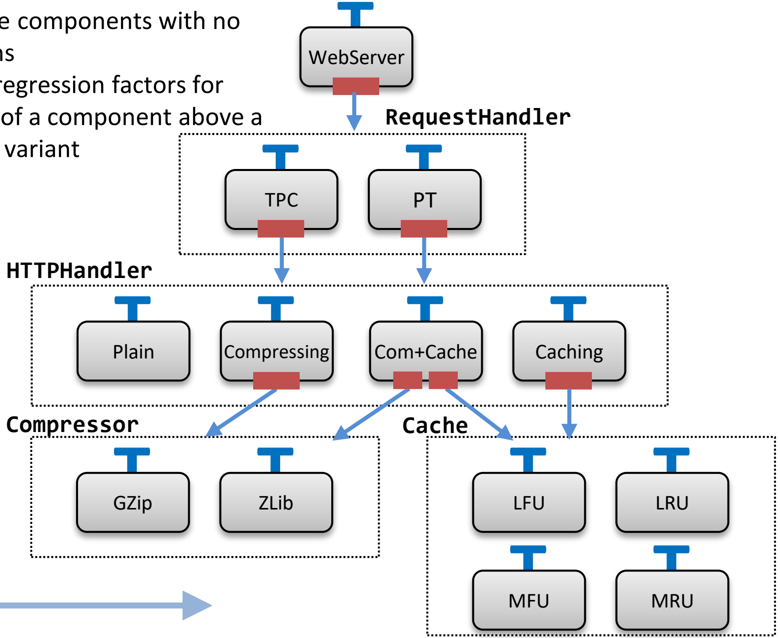
APPROACH IN DETAIL – LEARNING MODULE

- An **online learning algorithm** which helps solve the *search space explosion,* and balances *exploration* and *exploitation*

APPROACH IN DETAIL – LEARNING MODULE

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

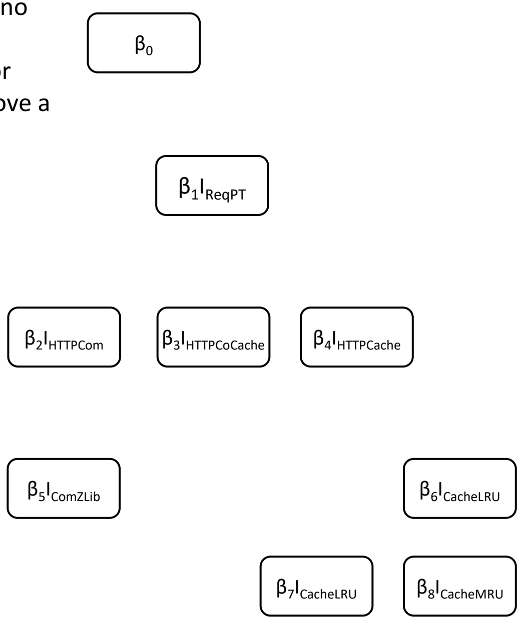
> remove components with no variations
> make regression factors for variants of a component above a baseline variant



Approach in detail – Learning module

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

> remove components with no variations
> make regression factors for variants of a component above a baseline variant



β_0

$\beta_1 I_{ReqPT}$

$\beta_2 I_{HTTPCom}$ $\beta_3 I_{HTTPCoCache}$ $\beta_4 I_{HTTPCache}$

$\beta_5 I_{ComZLib}$ $\beta_6 I_{CacheLRU}$

$\beta_7 I_{CacheLRU}$ $\beta_8 I_{CacheMRU}$

Approach in detail – Learning module

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

- β_0
- $\beta_1 I_{ReqPT}$
- $\beta_2 I_{HTTPCom}$
- $\beta_3 I_{HTTPCoCache}$
- $\beta_4 I_{HTTPCache}$
- $\beta_5 I_{ComZLib}$
- $\beta_6 I_{CacheLRU}$
- $\beta_7 I_{CacheLRU}$
- $\beta_8 I_{CacheMRU}$

TPC.Plain	**PT.Plain**	**TPC.Com.GZip**	**TPC.Com.ZLib**	**…**
β_0 1	β_0 1	β_0 1	β_0 1	…
β_1 0	β_1 1	β_1 0	β_1 0	…
β_2 0	β_2 0	β_2 1	β_2 1	…
β_3 0	β_3 0	β_3 0	β_3 0	…
β_4 0	β_4 0	β_4 0	β_4 0	…
β_5 0	β_5 0	β_5 0	β_5 1	…
β_6 0	β_6 0	β_6 0	β_6 0	…
β_7 0	β_7 0	β_7 0	β_7 0	…
β_8 0	β_8 0	β_8 0	β_8 0	…

Approach in detail – Learning module

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

β_0

$\beta_1 I_{ReqPT}$

$\beta_2 I_{HTTPCom}$

$\beta_3 I_{HTTPCoCache}$

$\beta_4 I_{HTTPCache}$

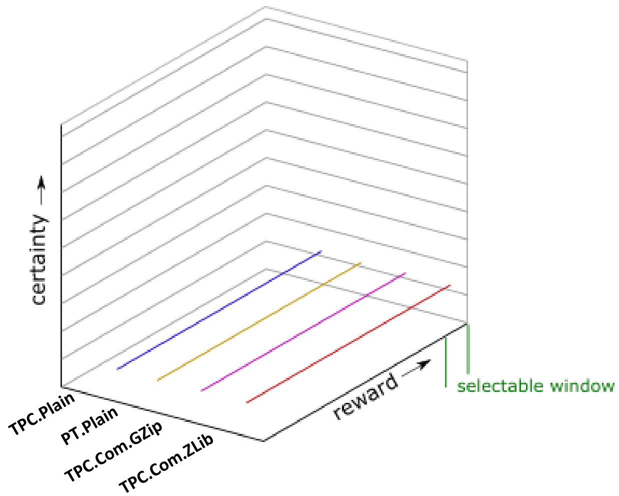
$\beta_5 I_{ComZLib}$

$\beta_6 I_{CacheLRU}$

$\beta_7 I_{CacheLRU}$

$\beta_8 I_{CacheMRU}$

TPC.Plain	**PT.Plain**	**TPC.Com.GZip**	**TPC.Com.ZLib**	**…**
β_0 1	β_0 1	β_0 1	β_0 1	…
β_1 0	β_1 1	β_1 0	β_1 0	…
β_2 0	β_2 0	β_2 1	β_2 1	…
β_3 0	β_3 0	β_3 0	β_3 0	…
β_4 0	β_4 0	β_4 0	β_4 0	…
β_5 0	β_5 0	β_5 0	β_5 1	…
β_6 0	β_6 0	β_6 0	β_6 0	…
β_7 0	β_7 0	β_7 0	β_7 0	…
β_8 0	β_8 0	β_8 0	β_8 0	…



every *n* seconds, select a composition that we either (1) know little about; or (2) know performs well

Approach in Detail – Learning module

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

β_0

$\beta_1 I_{ReqPT}$

$\beta_2 I_{HTTPCom}$

$\beta_3 I_{HTTPCoCache}$

$\beta_4 I_{HTTPCache}$

$\beta_5 I_{ComZLib}$

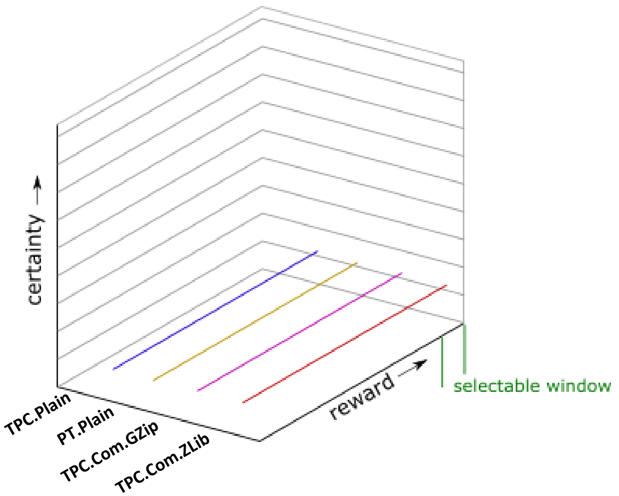
$\beta_6 I_{CacheLRU}$

$\beta_7 I_{CacheLRU}$

$\beta_8 I_{CacheMRU}$

TPC.Plain	PT.Plain	TPC.Com.GZip	TPC.Com.ZLib	...
β_0 1	β_0 1	β_0 1	β_0 1	...
β_1 0	β_1 1	β_1 0	β_1 0	...
β_2 0	β_2 0	β_2 1	β_2 1	...
β_3 0	β_3 0	β_3 0	β_3 0	...
β_4 0	β_4 0	β_4 0	β_4 0	...
β_5 0	β_5 0	β_5 0	β_5 1	...
β_6 0	β_6 0	β_6 0	β_6 0	...
β_7 0	β_7 0	β_7 0	β_7 0	...
β_8 0	β_8 0	β_8 0	β_8 0	...

select



Approach in detail – Learning module

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

β_0

$\beta_1 I_{ReqPT}$

$\beta_2 I_{HTTPCom}$

$\beta_3 I_{HTTPCoCache}$

$\beta_4 I_{HTTPCache}$

$\beta_5 I_{ComZLib}$

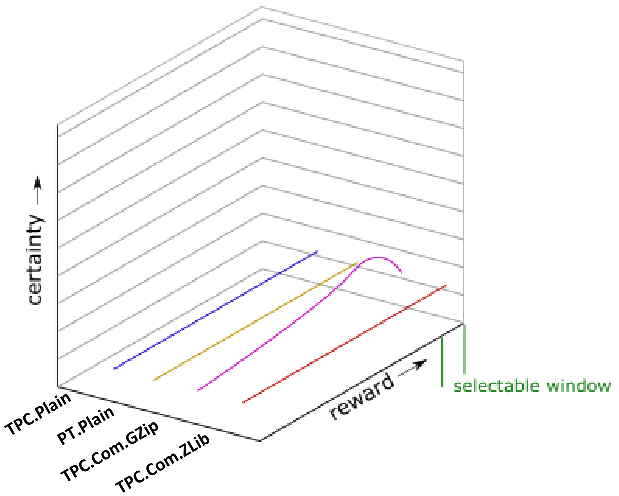
$\beta_6 I_{CacheLRU}$

$\beta_7 I_{CacheLRU}$

$\beta_8 I_{CacheMRU}$

TPC.Plain	PT.Plain	TPC.Com.GZip	TPC.Com.ZLib	...
β_0 1	β_0 1	β_0 1	β_0 1	...
β_1 0	β_1 1	β_1 0	β_1 0	...
β_2 0	β_2 0	β_2 1	β_2 1	...
β_3 0	β_3 0	β_3 0	β_3 0	...
β_4 0	β_4 0	β_4 0	β_4 0	...
β_5 0	β_5 0	β_5 0	β_5 1	...
β_6 0	β_6 0	β_6 0	β_6 0	...
β_7 0	β_7 0	β_7 0	β_7 0	...
β_8 0	β_8 0	β_8 0	β_8 0	...

select



observe reward, update estimates

Approach in detail – Learning module

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

β_0

$\beta_1 I_{ReqPT}$

$\beta_2 I_{HTTPCom}$

$\beta_3 I_{HTTPCoCache}$

$\beta_4 I_{HTTPCache}$

$\beta_5 I_{ComZLib}$

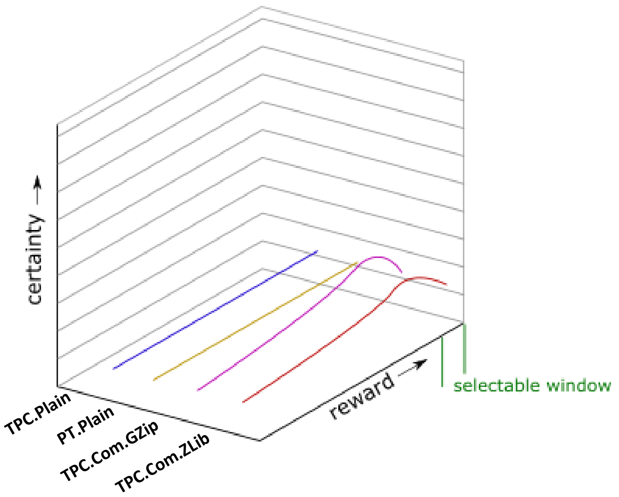
$\beta_6 I_{CacheLRU}$

$\beta_7 I_{CacheLRU}$

$\beta_8 I_{CacheMRU}$

TPC.Plain	PT.Plain	TPC.Com.GZip	TPC.Com.ZLib	…
β_0 1	β_0 1	β_0 1	β_0 1	…
β_1 0	β_1 1	β_1 0	β_1 0	…
β_2 0	β_2 0	β_2 1	β_2 1	…
β_3 0	β_3 0	β_3 0	β_3 0	…
β_4 0	β_4 0	β_4 0	β_4 0	…
β_5 0	β_5 0	β_5 0	β_5 1	…
β_6 0	β_6 0	β_6 0	β_6 0	…
β_7 0	β_7 0	β_7 0	β_7 0	…
β_8 0	β_8 0	β_8 0	β_8 0	…

select



share estimates across configurations

Approach in detail – Learning module

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

β_0

$\beta_1 I_{ReqPT}$

$\beta_2 I_{HTTPCom}$

$\beta_3 I_{HTTPCoCache}$

$\beta_4 I_{HTTPCache}$

$\beta_5 I_{ComZLib}$

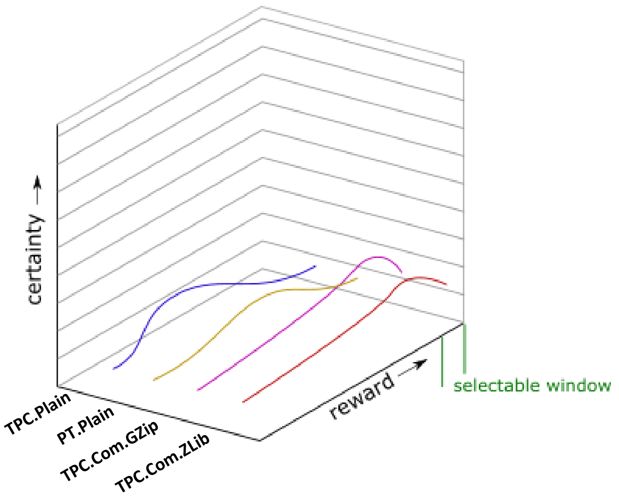
$\beta_6 I_{CacheLRU}$

$\beta_7 I_{CacheLRU}$

$\beta_8 I_{CacheMRU}$

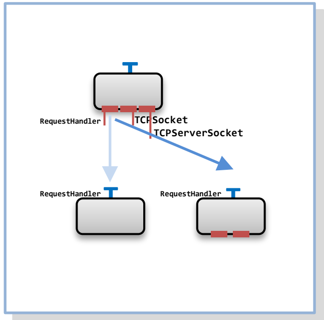
TPC.Plain	PT.Plain	TPC.Com.GZip	TPC.Com.ZLib	...
β_0 1	β_0 1	β_0 1	β_0 1	...
β_1 0	β_1 1	β_1 0	β_1 0	...
β_2 0	β_2 0	β_2 1	β_2 1	...
β_3 0	β_3 0	β_3 0	β_3 0	...
β_4 0	β_4 0	β_4 0	β_4 0	...
β_5 0	β_5 0	β_5 0	β_5 1	...
β_6 0	β_6 0	β_6 0	β_6 0	...
β_7 0	β_7 0	β_7 0	β_7 0	...
β_8 0	β_8 0	β_8 0	β_8 0	...

select

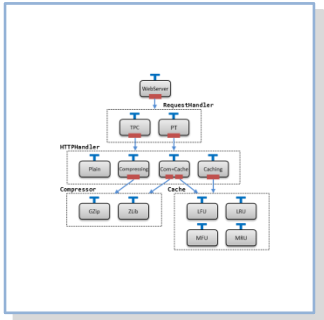


continue learning…

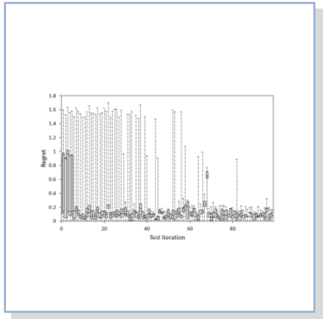
EVALUATION



Adaptation speed



Performance ground truth



Learning characteristics

Evaluation – learning characteristics

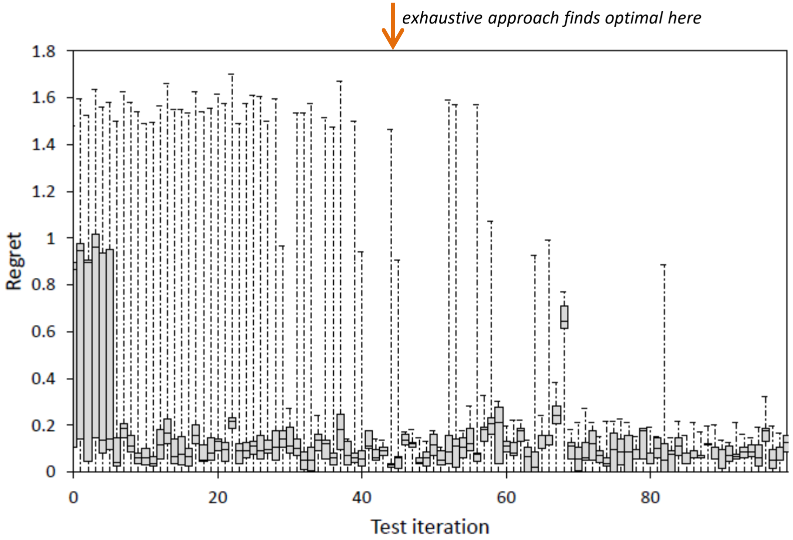
- Using our web server as an example, we evaluate how optimal systems emerge over time in our approach

- We use a set of different workloads (client request patterns), including synthetic and real-world traces

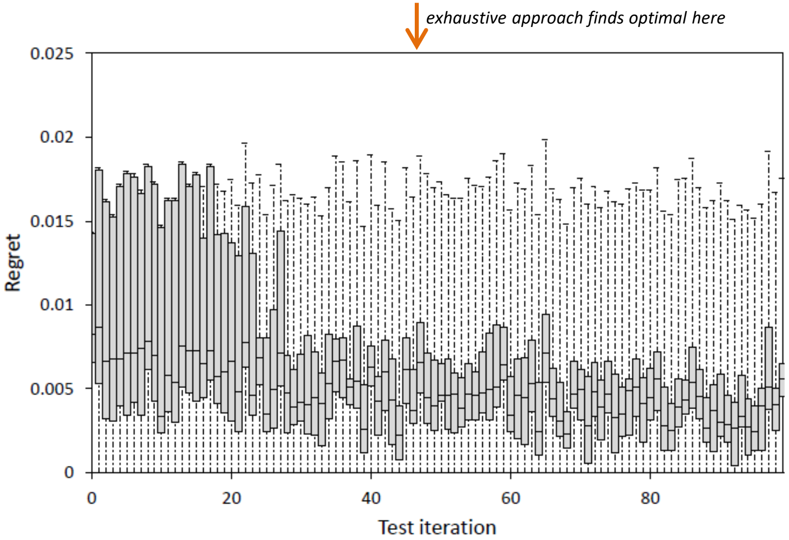
- The only data available for learning is **(i)** the configuration set; and **(ii)** the metrics and events that the system emits (in this case, response times and request types/volumes)

EVALUATION – LEARNING CHARACTERISTICS

Key result: convergence on optimal solution happens much faster than exhaustive search



Workload: small text files

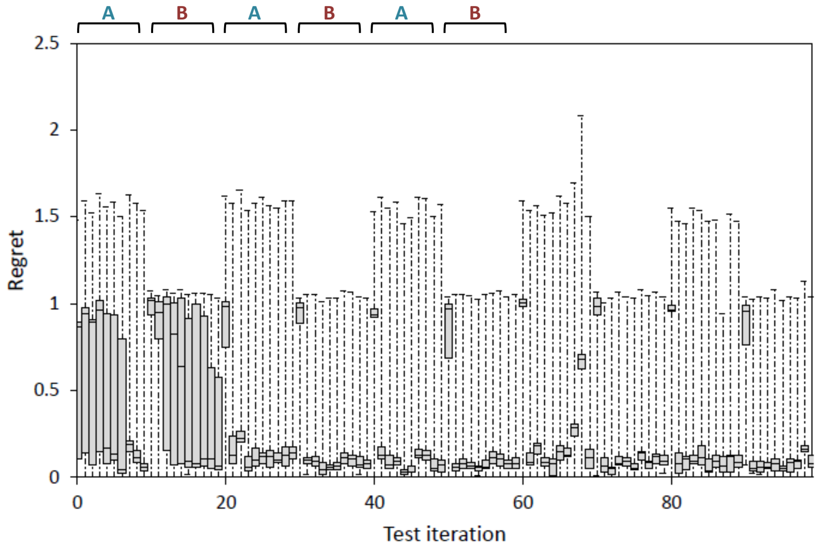


Workload: large image files

Distance from optimal solution over time, averaged across 1,000 runs

Evaluation – learning characteristics

Key result: once learned, workload changes are rapidly detected and adjusted to

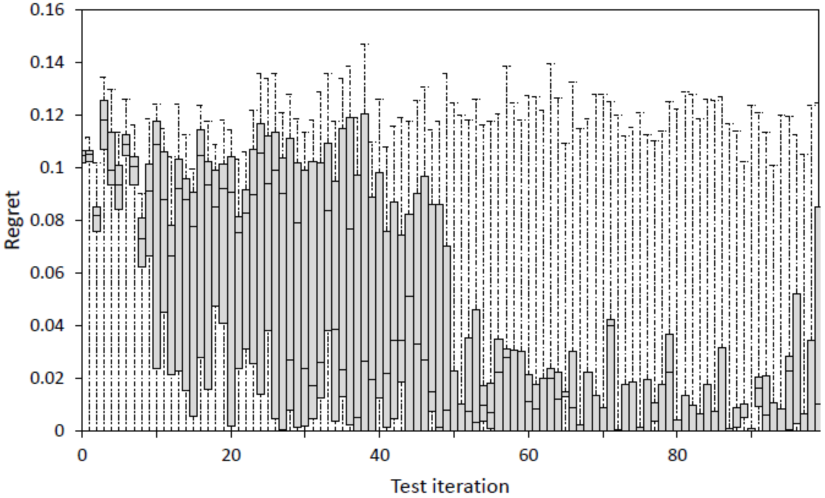


Workload: cycling between two different request patterns every 100 seconds

Distance from optimal solution over time, averaged across 1,000 runs

EVALUATION – LEARNING CHARACTERISTICS

Key result: convergence occurs in a highly varying real-world workload trace



Workload: real-world workload taken from a publicly available NASA web server trace

Distance from optimal solution over time, averaged across 1,000 runs

EVALUATION – OTHER INSIGHTS

- More broadly, *unexpected* optimal designs that emerged due to machine learning were some of the most interesting results

- This highlighted cases in which our assumptions were wrong about how a given composition would behave, or examples of programmer error / poor design choices

Summary

- Presented the idea of *emergent software systems* as a new solution to system complexity and deployment dynamics

- We use a paradigm of continuous self-assembly, finding optimal systems via automated composition from small building blocks

- Future work: studying more applications (other server types, AI, robotics); automated generation of variants; automated environment classification; distributed emergent systems (e.g. entire datacentre software landscapes)

- download our code at http://www.projectdana.com -

the authors

Barry
Porter

Matthew
Grieves

Roberto
Rodrigues Filho

David
Leslie