EMERGENT SOFTWARE SYSTEMS

Summer School

Barry Porter & Roberto Rodrigues Filho
School of Computing and Communications
Lancaster University

Funded by The Royal Society Newton Fund

THE ROYAL SOCIETY

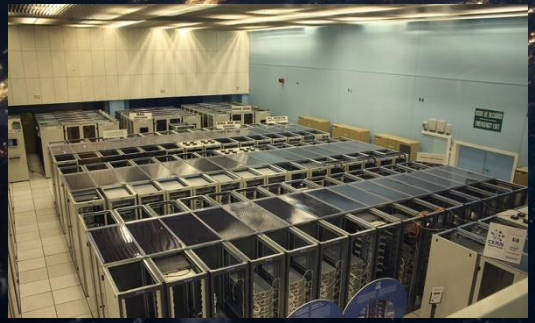
Emergent Software Systems

- Case study system
  - Data centres and web servers
  - Complexity, dynamism and variation
  - Learning dimensions

- Challenges
  - Although the ESS concept is simple from the developer perspective, there are hard problems within the ESS framework

An ESS Case Study

Emergent Web Server

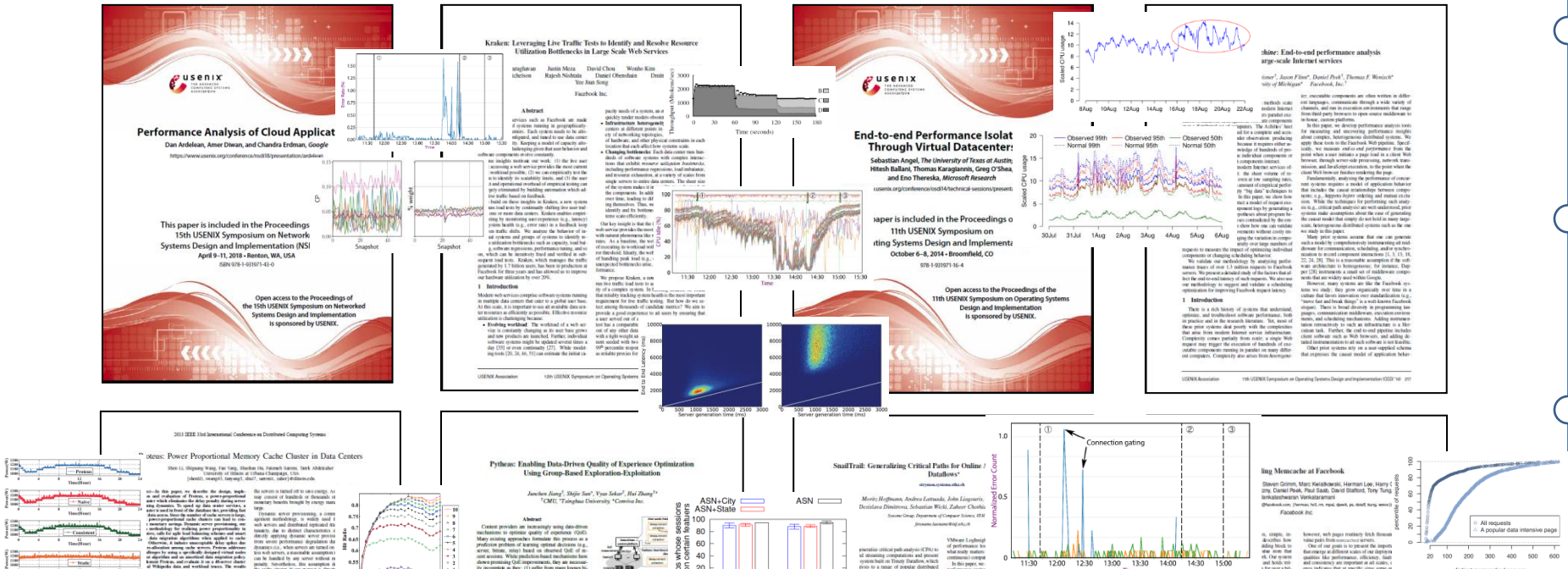


```
GET / HTTP/1.1
Host: www.lancaster.ac.uk
User-Agent: Mozilla/5.0 (Windows NT 10.0) Firefox/63.0
Accept: text/html,application/xhtml+xml,application
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

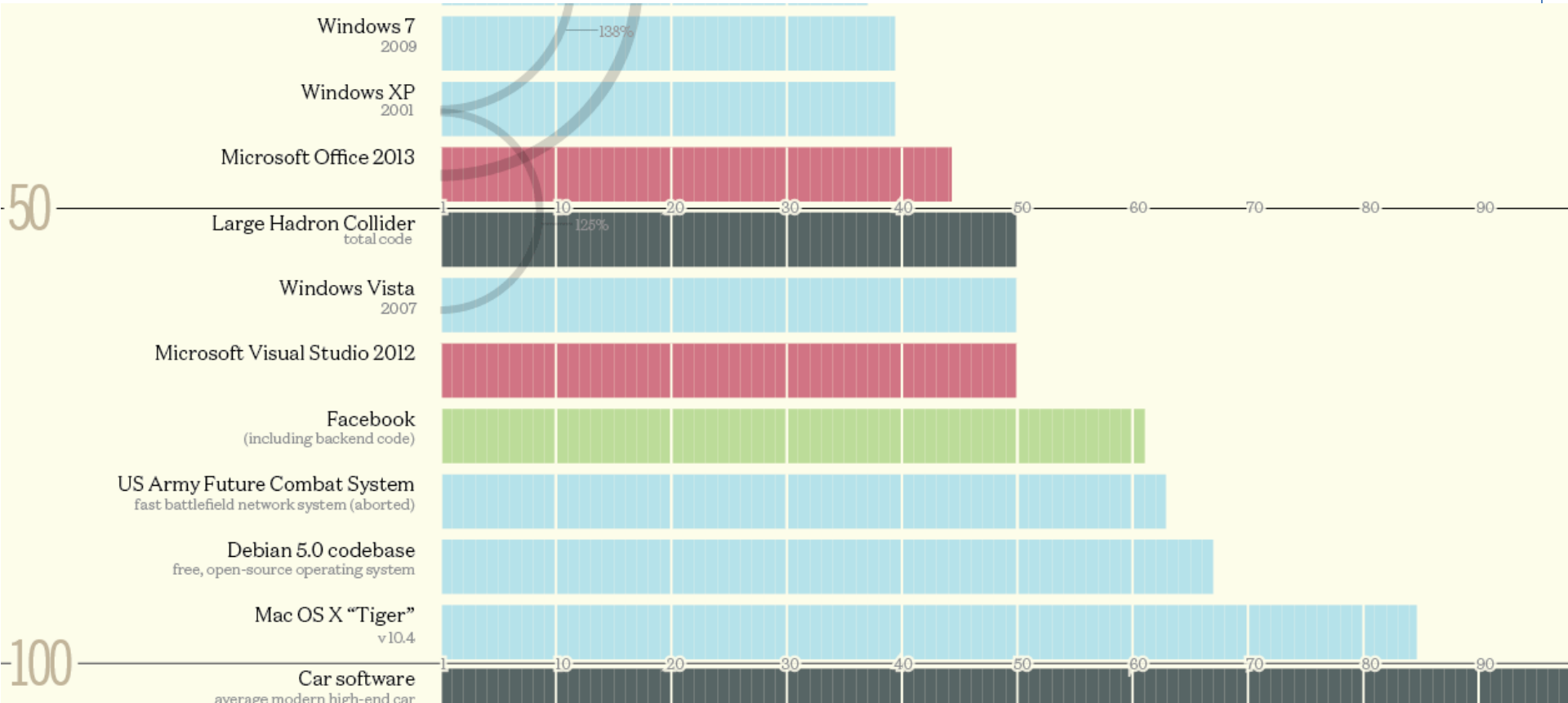
www.lancaster.ac.uk

Emergent Web Server

- While “simple”, building efficient web serving infrastructure is highly challenging even for the largest companies



Emergent Web Server



Emergent Web Server

- They’re challenging to build because the code base is increasingly complex, with server-side scripting and multi-source content fusion
- In addition, the deployment environment is highly dynamic and very hard to predict in advance
- Optimising this kind of system continues to be a major research challenge in both industry and academia

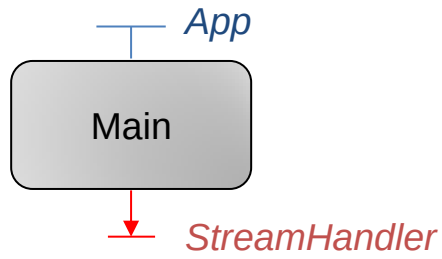
Emergent Web Server

- We start by defining a basic architecture, broken down into reusable components (building blocks)

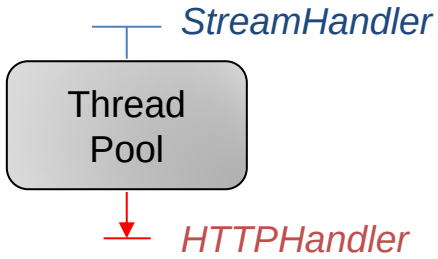
- This can be any initial architecture we like; we can change it later in collaboration with what the ESS observes

Emergent Web Server

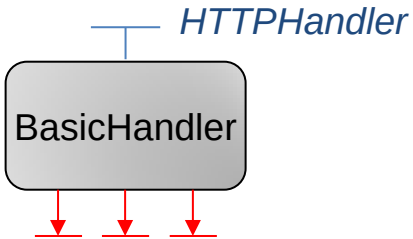
Open TCP master socket; accept new client connections



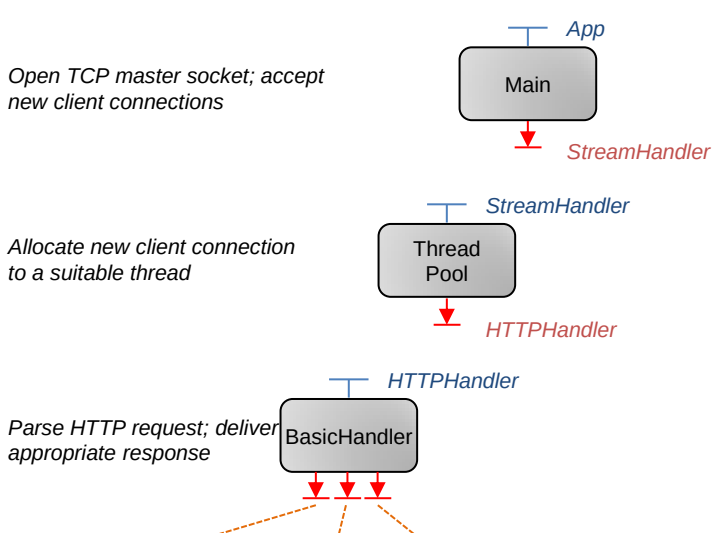
Allocate new client connection to a suitable thread



Parse HTTP request; deliver appropriate response



Emergent Web Server

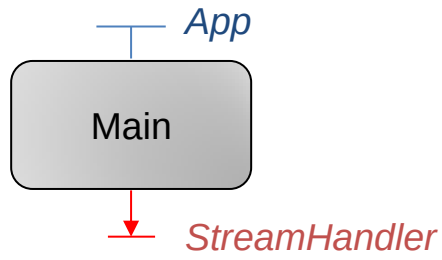


New custom code

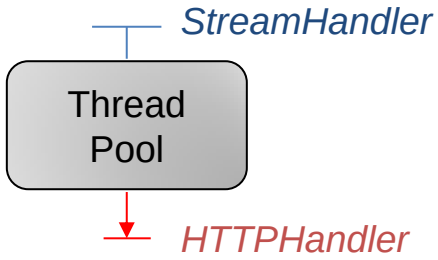
Standard Library (pre-made variations)

Emergent Web Server

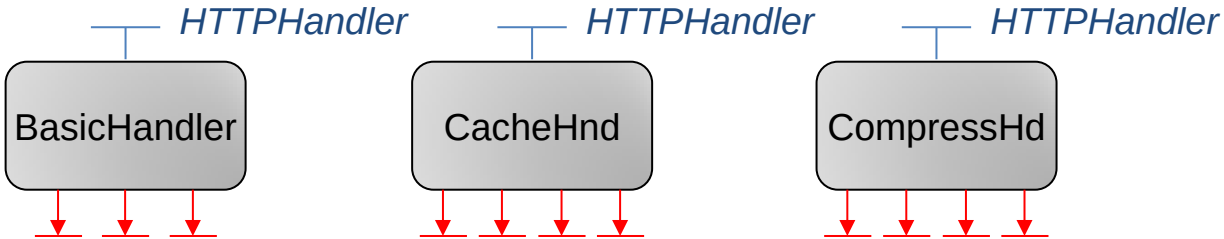
Open TCP master socket; accept new client connections



Allocate new client connection to a suitable thread

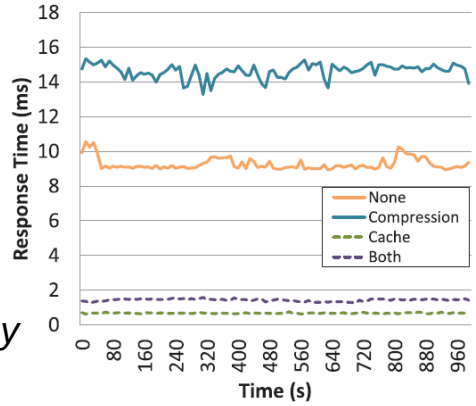


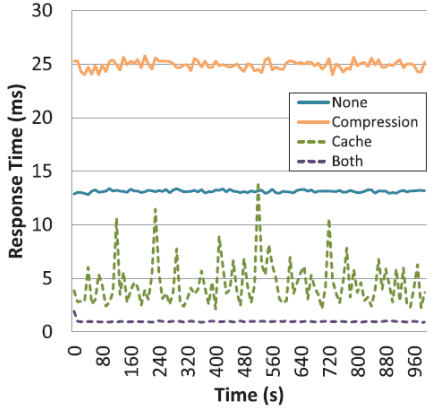
Parse HTTP request; deliver appropriate response



Emergent Web Server

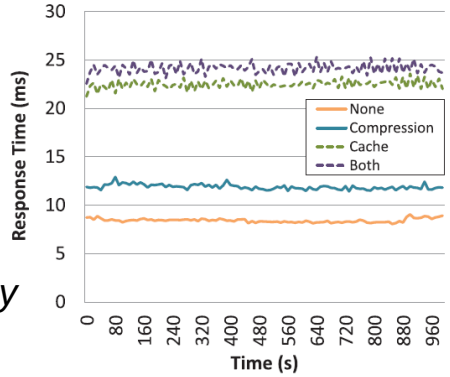
Mostly text, low entropy

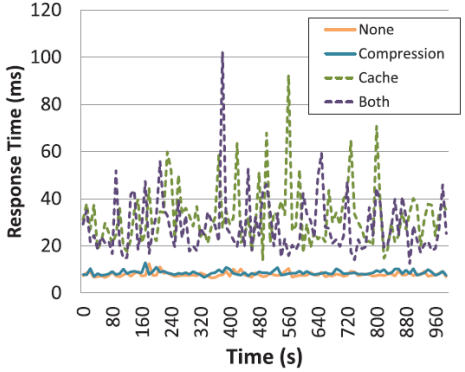




Mostly image, low entropy

Mostly text, high entropy





NASA web trace

Emergent Web Server – Learning

- We have a collection of building blocks and variations (both custom-built and in the standard library)
- Machine learning can automatically discover all valid compositions of behaviour which result in a web server
- However, the learning agent has no sense of *direction* about which behavioural compositions are better or worse

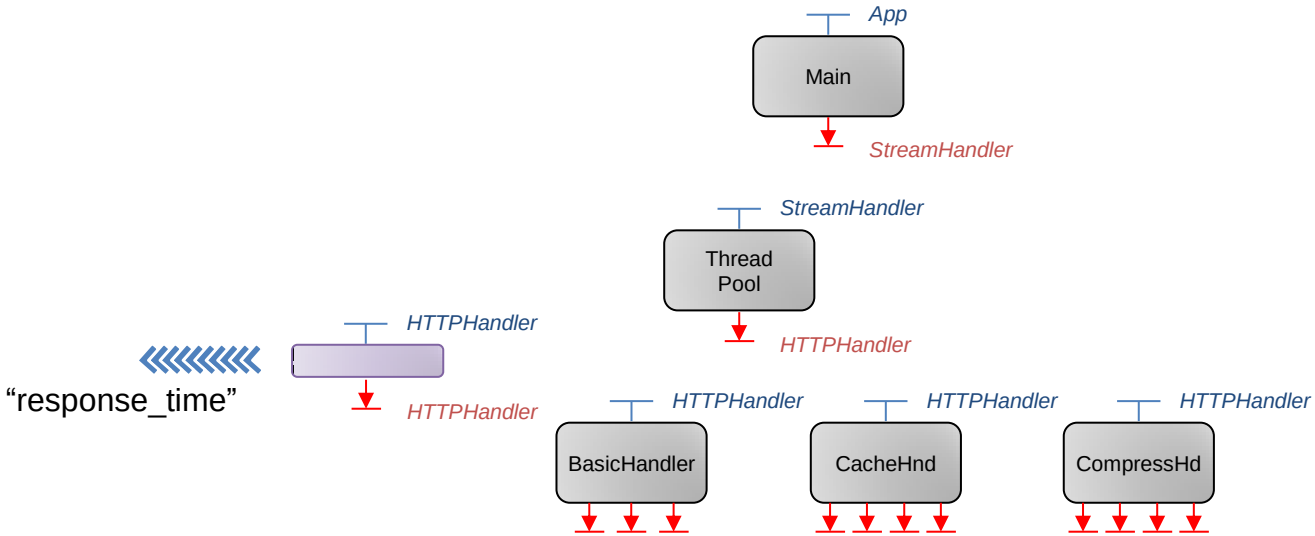
Emergent Web Server – Learning

- Next we need to define data streams which inform decision making for real-time machine learning

- These come in two flavours: **metrics** which explain how the system is behaving and define our reward levels; and **events** which explain how the deployment environment looks so that we can try to correlate different behaviours against different environment characteristics

Emergent Web Server – Learning

- We can define a **metric** as *response time* in our web server, and we can automatically insert a proxy component to measure this response time

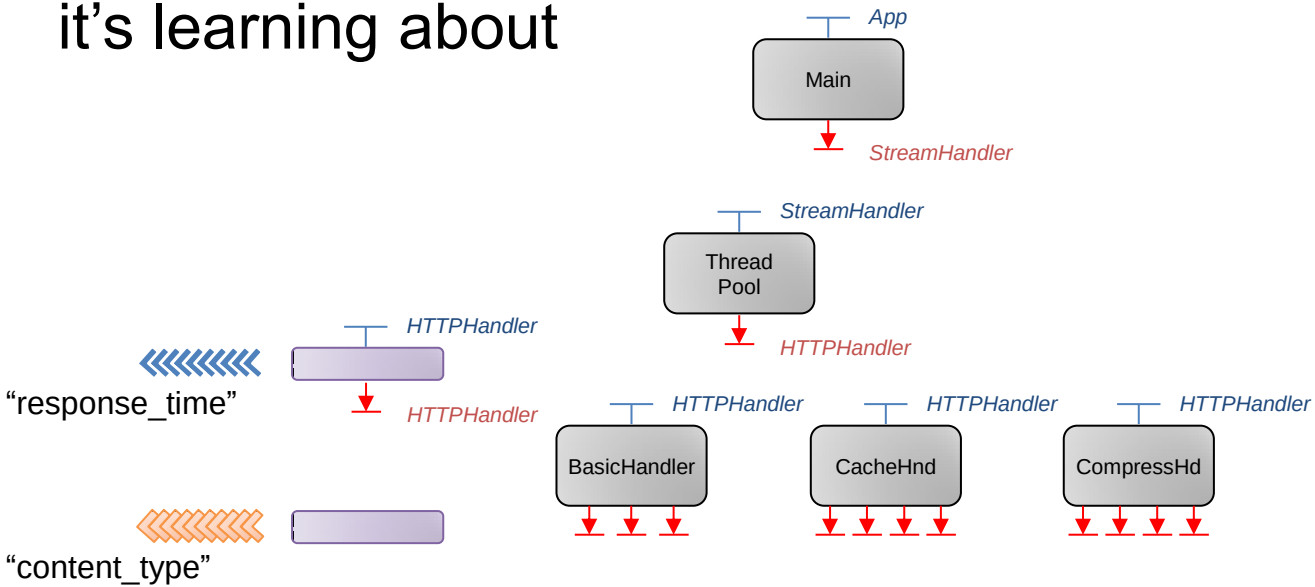


Emergent Web Server – Learning

- We can define an **event** as a content type, which may help to explain different behavioural characteristics
  - We could define further events as hardware characteristics such as current CPU load, memory occupancy, average latency, etc.
- This requires a little more custom code in a monitoring proxy to extract content types from requests, but is very simple to create

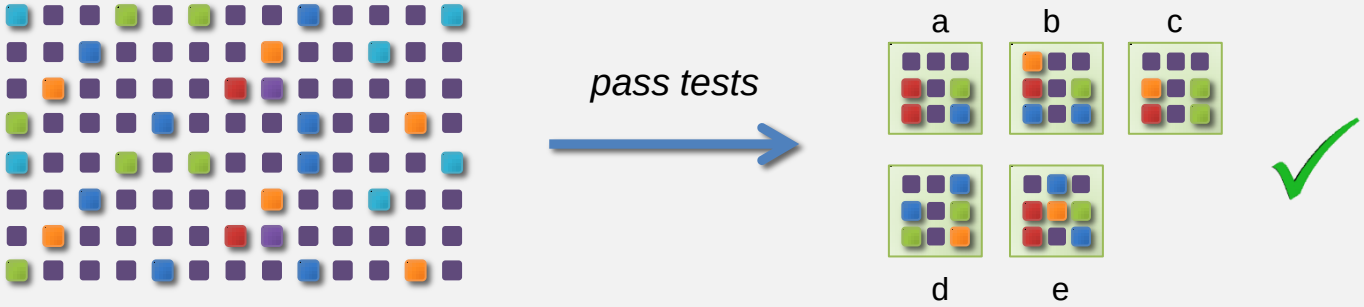
Emergent Web Server – Learning

- Both streams of data are labelled and fed into our ESS framework; the framework itself **does not need to know** anything about what the labels mean, or about the system it's learning about

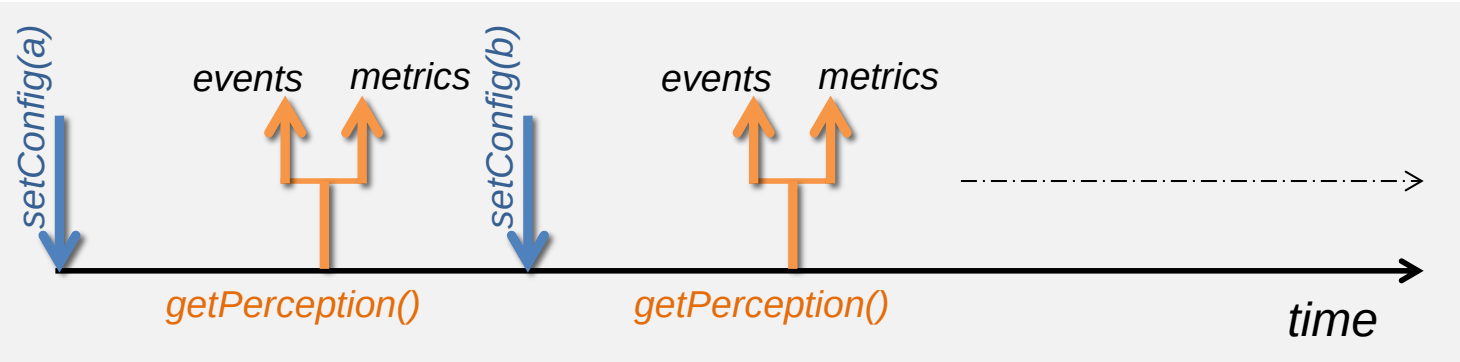


Emergent Software

offline

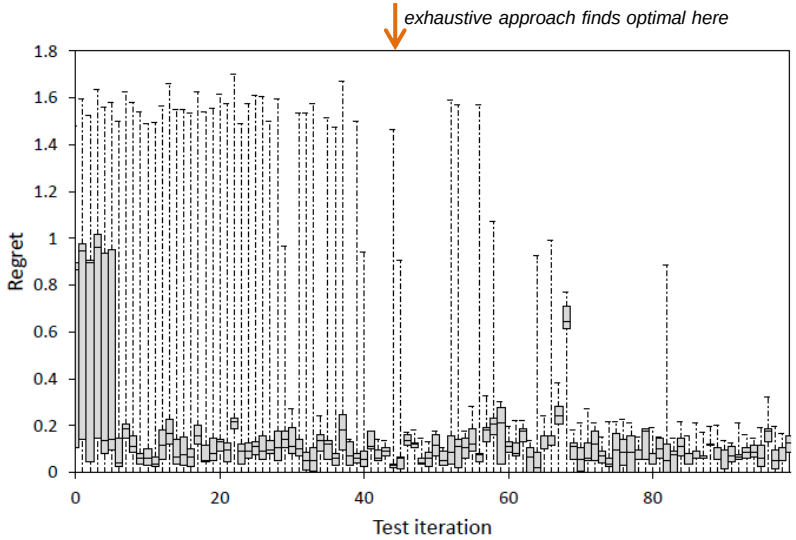


online

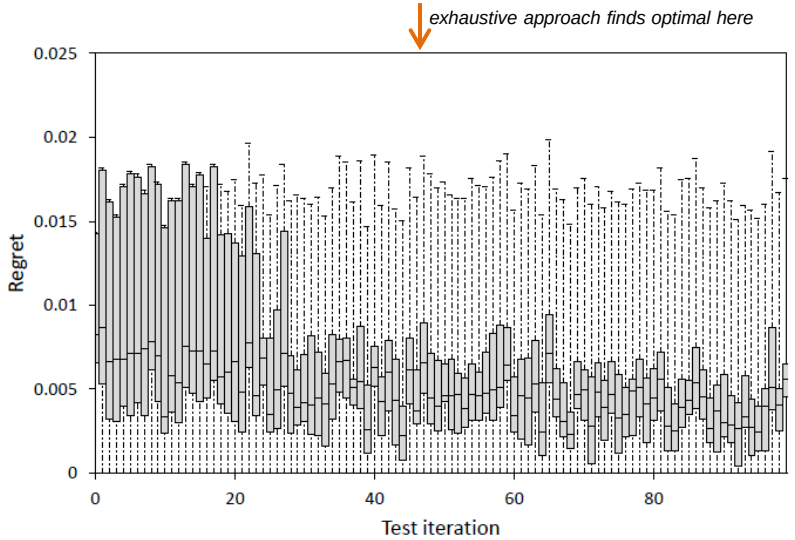


Emergent Web Server – Learning

- Real-time machine learning can quickly explore a carefully chosen set of behavioural compositions to learn about which broad kinds of behaviour are best…



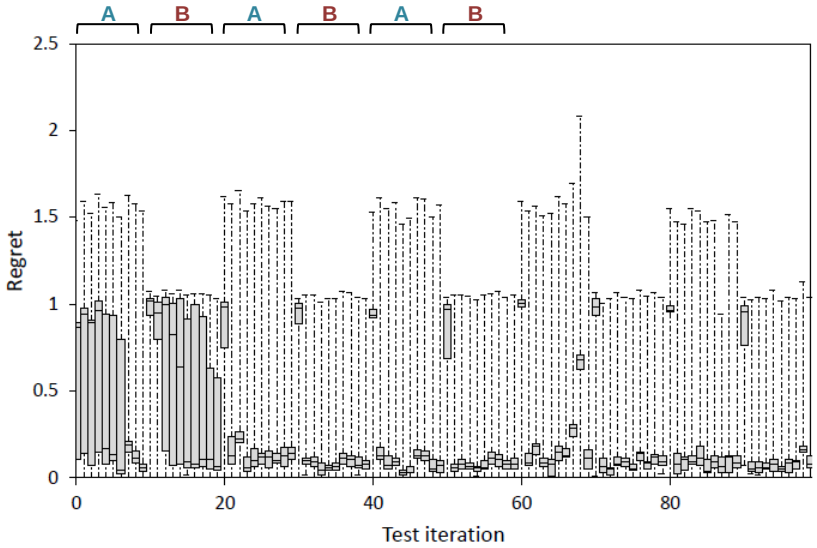
Workload: small text files



Workload: large image files

Emergent Web Server – Learning

- It can also learn about different classes of environment, even though it only has abstractly labelled data



Emergent Web Server – Summary

- Define any custom code needed for the new system, arranged in re-usable components
  - Link these against existing general-purpose interfaces in the standard library where needed / appropriate
- Create variations of behaviour that may be useful
- Define **metrics** and **events** for machine learning to reason about *reward* and *environment*

Challenges

Major Challenges in ESS frameworks

- Creating emergent systems is designed to make life easier for the system developer by pushing more responsibility into automated processes / ML

- However, to do this there are major challenges in the design of the ESS framework which assembles a target system and learns how to control its behaviour over time

Major Challenges in ESS frameworks

- Creating useful variation for divergence
- Real-time learning with relative performance
- Search space complexity
- Self-referential fitness landscapes
- Abstracting and reasoning about the environment
- Data quality and perception errors
- Fusing online and offline learning
- Application to critical systems
- Developer interaction

Major Challenges in ESS frameworks

- Creating useful variation for divergence

- We expect developers to create behavioural variations which cover interesting parts of a fitness landscape
  - In some cases this means “guessing” which kinds of behaviour might be useful in certain environment ranges

- In future it’s much more appealing to have machine processes automatically create variations
  - We’ve done some initial work in this area, but it’s a hard problem

Experiments in Genetic Divergence for Emergent Systems, Christopher McGowan, Alexander Wild, Barry Porter, *International Genetic Improvement Workshop*, June 2018

Major Challenges in ESS frameworks

- Real-time learning with relative performance

- In emergent systems we want to learn how a system **actually** behaves in the deployment environment to which it is really subjected at runtime
  - This avoids problems with humans trying to predict runtime effects, which has been regularly shown to have high error rates

- The best way to do this is by designing purely real-time learning which starts (in a production environment) from zero information and so is forced to learn quickly
  - However, because the system has no baseline for performance, it is constantly unsure if the performance of an un-tried composition might be better
  - The environment also behaves erratically: we often don't have the ability to directly compare two different compositions in sequence in the same environment

Major Challenges in ESS frameworks

- Search space complexity

- As we increase the scale of software systems, an emergent system becomes a very large search space of possible behavioural permutations

- Because we do all learning online, in response to actually observed conditions, we have to somehow navigate these search spaces very quickly to respond in reasonable time to new environments

Major Challenges in ESS frameworks

- Self-referential fitness landscapes

- As we explore different behavioural compositions online, we can sometimes observe a strange effect: changing a composition can appear to change the environment

- This can be particularly acute if event or metric types are chosen poorly, such as reporting the average file size being requested but measuring this as the compressed size when a compression behaviour is in use
  - Our learning system then needs to somehow understand that an environment change has a causal connection to a composition change

Major Challenges in ESS frameworks

- Abstracting and reasoning about the environment

- A data trace about the deployment environment, particularly around elements such as the current request pattern, will rarely explain classes of environment which correlate to optimal behaviours
  - An example being capture of media types in a web server, when the important detail is the level of entropy in requests, or the compressibility of the data

- Environment classifiers may therefore need to make complex inferences to construct new classes from the raw data stream which show the actual correlations between environment and behaviour
  - Online classification of this kind remains a hard problem in ML research

Major Challenges in ESS frameworks

- Data quality and perception errors

- Consider a situation in which we've discovered the best action for a classified environment E, and then suddenly the performance of that action changes radically without an observable change in the environment

- This is a "perception error": there is something about the environment that we don't have as an available data stream and so cannot observe the reason for changes
  - Although we can automatically report that this has happened, we cannot easily correct it without human assistance

Major Challenges in ESS frameworks

- Fusing online and offline learning

- Because our online learning systems must learn as they encounter new environments, and environments can change without warning, it can be a real challenge to build up enough information about how the system behaves in different compositions for a given environment

- A solution to this may be to fuse online learning with a limited form of offline learning, where we replay observed environment effects offline to gain a broader understanding of different behavioural permutations in a more controlled setting
  - However, we don't want to have the run the entire system offline as we would need to duplicate all of the resources which may not be feasible, so we need approaches to approximate these conditions in more limited offline tests

Major Challenges in ESS frameworks

- Application to critical systems

- To date we have applied to ESS concept to “soft” systems in which it is safe to explore different behaviours without causing any damage

- If we look towards critical systems, like power station control or flight control, we have different requirements around the acceptable extent of behavioural exploration
  - However, it is also in these kinds of systems that we **want** the software to find creative solutions to unexpected scenarios, which creates a kind of paradox in whether we allow exploration and when

Major Challenges in ESS frameworks

- Interaction with human developers

- Emergent systems are designed to ease the complexity of building large-scale software

- However, we still want a role for human engineers to interact with these systems at a higher level
  - By directing their overall trajectory, making creative suggestions of likely solutions, or ensuring their behaviour remains reasonable

- How we express this bi-directional interaction with humans at a sufficiently high level is an interesting challenge which could help to redefine what it means to be a software engineer

Summary

- We've seen a walk-through of a real emergent system (we'll use this system in upcoming practical sessions)

- We looked at some of the key challenges around designing the ESS control systems themselves, particularly around real-time machine learning