Scalable Applications

Design - Refactor - Host

Emanuele Rocca

Vrije Universiteit Amsterdam
Advanced Topics in Distributed Systems

October 14, 2011

Why scaling?

30 thousand hits on the web application in one day!

30 thousand potential new users!

Three different scenarios

1. scale from your garage (again)
2. scale an existing application
3. host applications in a scalable way

Outline

1. Garage Innovators
2. Keep the relational model
3. PaaS providers

Who is this Garage Innovator, again?

- good idea
- no money

Utility Computing

"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility. The computer utility could become the basis of a new and important industry."

– John McCarthy, MIT Centennial, 1961

Utility computing makes it feasable

The application should:

- scale up within seconds
- scale down when the crowd leaves (remember: no money)

Toolbox

- Storage Delivery Networks
- HTTP redirection
- Load balancers (network vs. application level)
- Round robin DNS

Houston, we have a problem

The database

But YOU can handle a flash crowd

10k new users in one day
with 150$

Outline

1 Garage Innovators

2 Keep the relational model

3 PaaS providers

Give me my RDBMS back!

- My data is relational
- I want strong consistency
- The application needs transactions
- Resistance to change

Access patterns?

Turns out that different access patterns allow for different optimizations.

- replication if you have lots of reads
- partitioning if you have writes that do not span partitions

Service-oriented modeling

Go for a *Service Oriented Design*.

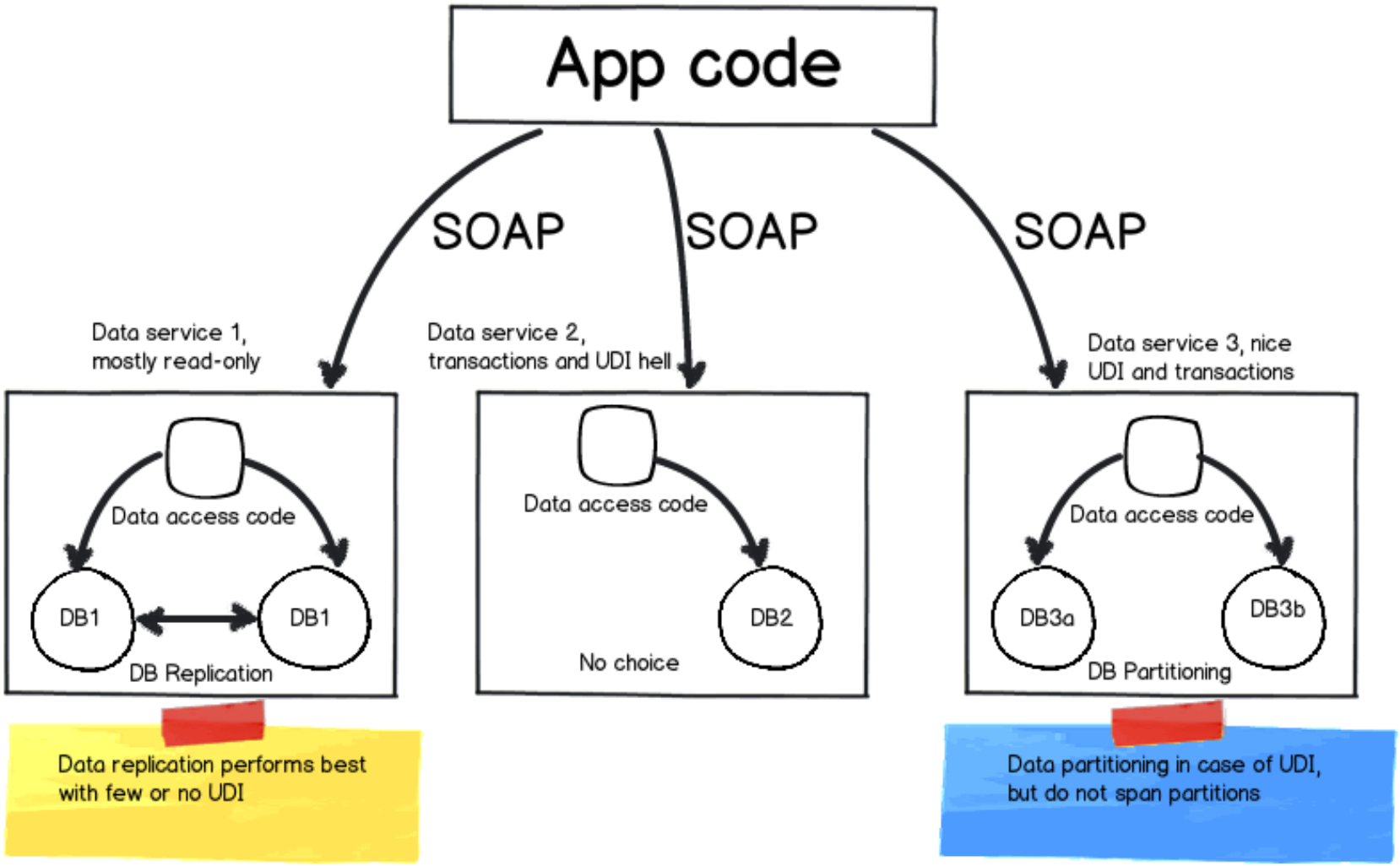
- separation of concerns
- independent data services
- each data service has exclusive access to its private data store

Service-oriented modeling

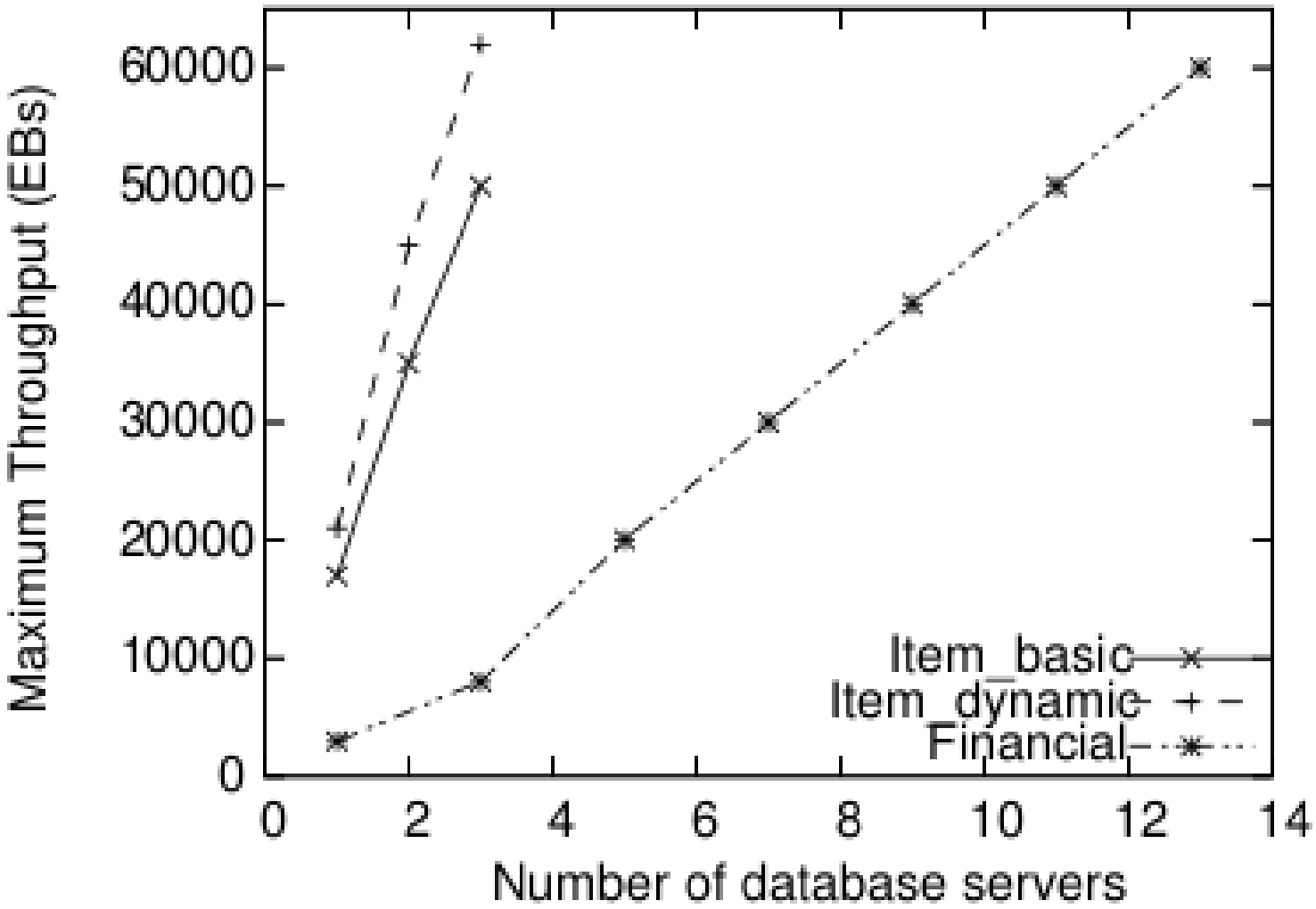
Go for a *Service Oriented Design*.

- separation of concerns
- independent data services
- each data service has exclusive access to its private data store
- simplification of each database workload
- more efficient use of classic scaling techniques

Service Oriented Data Denormalization



Scalability of individual services



Putting It All Together

- You have to know what you are doing
- Any update to the code may require to restructure your data

- Denormalization allows efficient use of scalability techniques
- Strong consistency and transactional properties

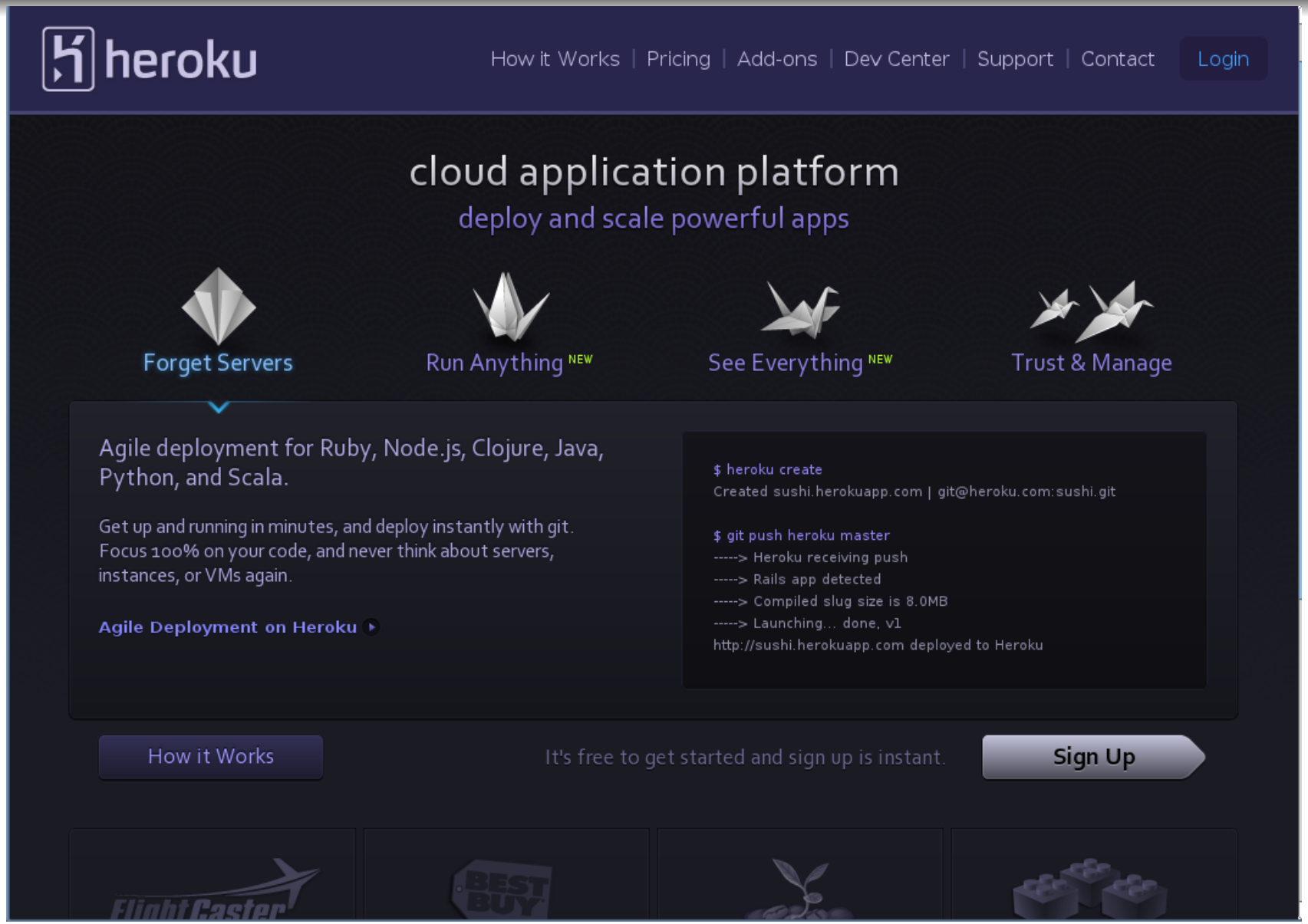
Outline

1 Garage Innovators

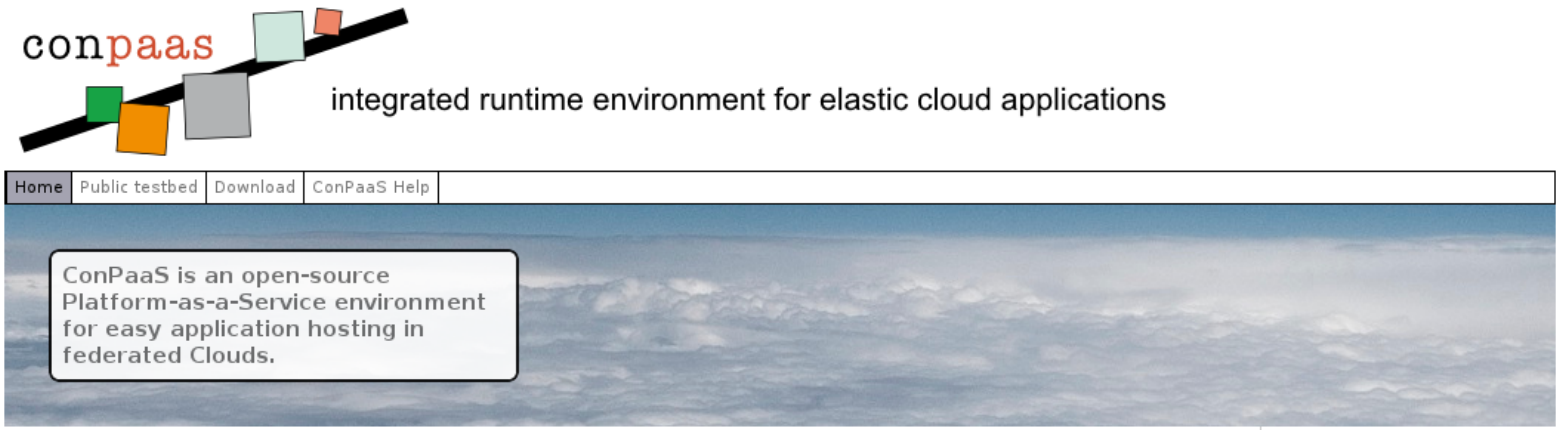
2 Keep the relational model

3 PaaS providers

We want to host applications in a scalable way



We want to host applications in a scalable way



PaaS providers have their amount of fun too

- cope with dynamic, changing workloads
- choose between different VM adaptations
  1. capacity control
  2. migration
  3. replication
- provide isolation
- fulfill SLAs

PaaS providers have their amount of fun too

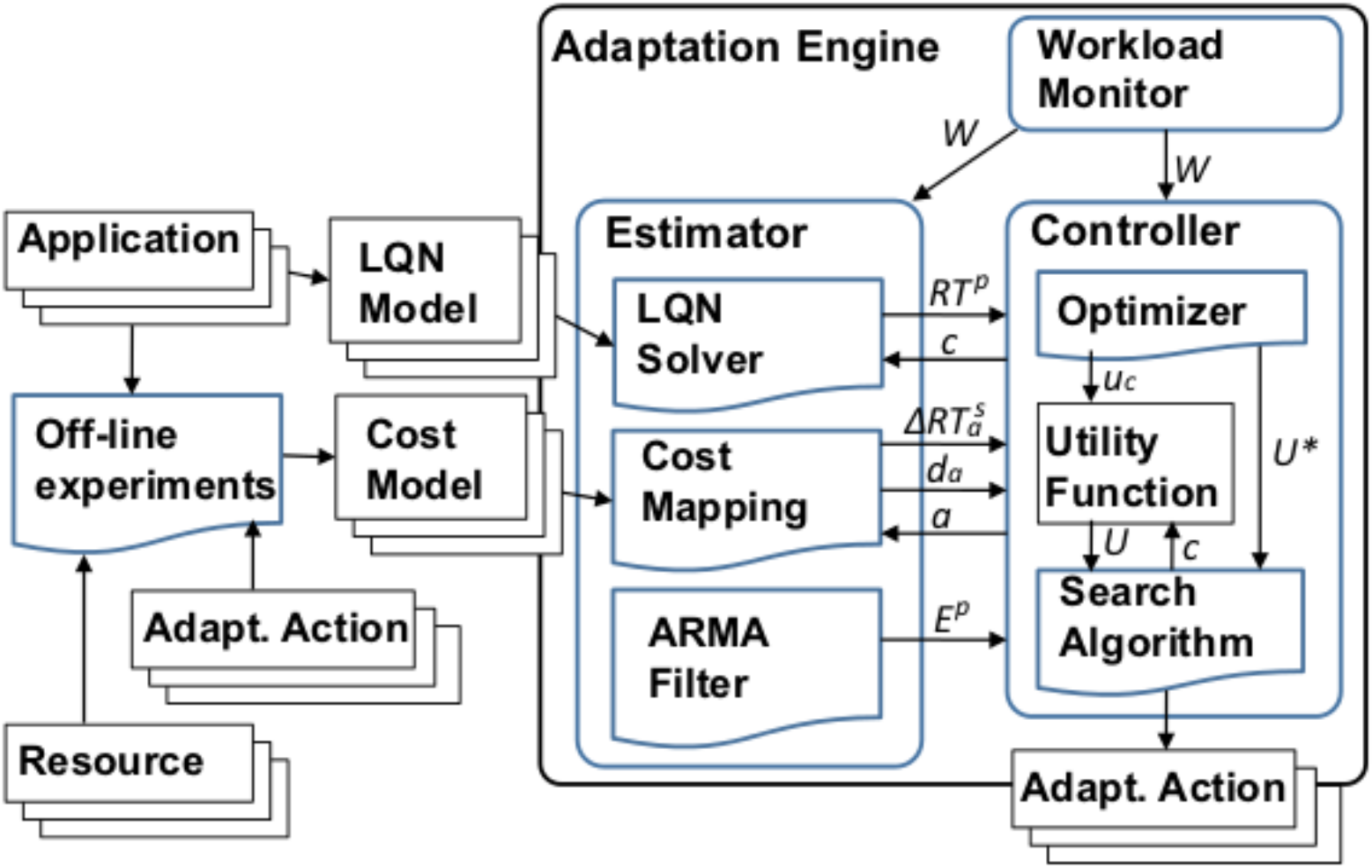
- cope with dynamic, changing workloads
- choose between different VM adaptations
  1. capacity control
  2. migration
  3. replication
- provide isolation
- fulfill SLAs

Changing the application is out of the question

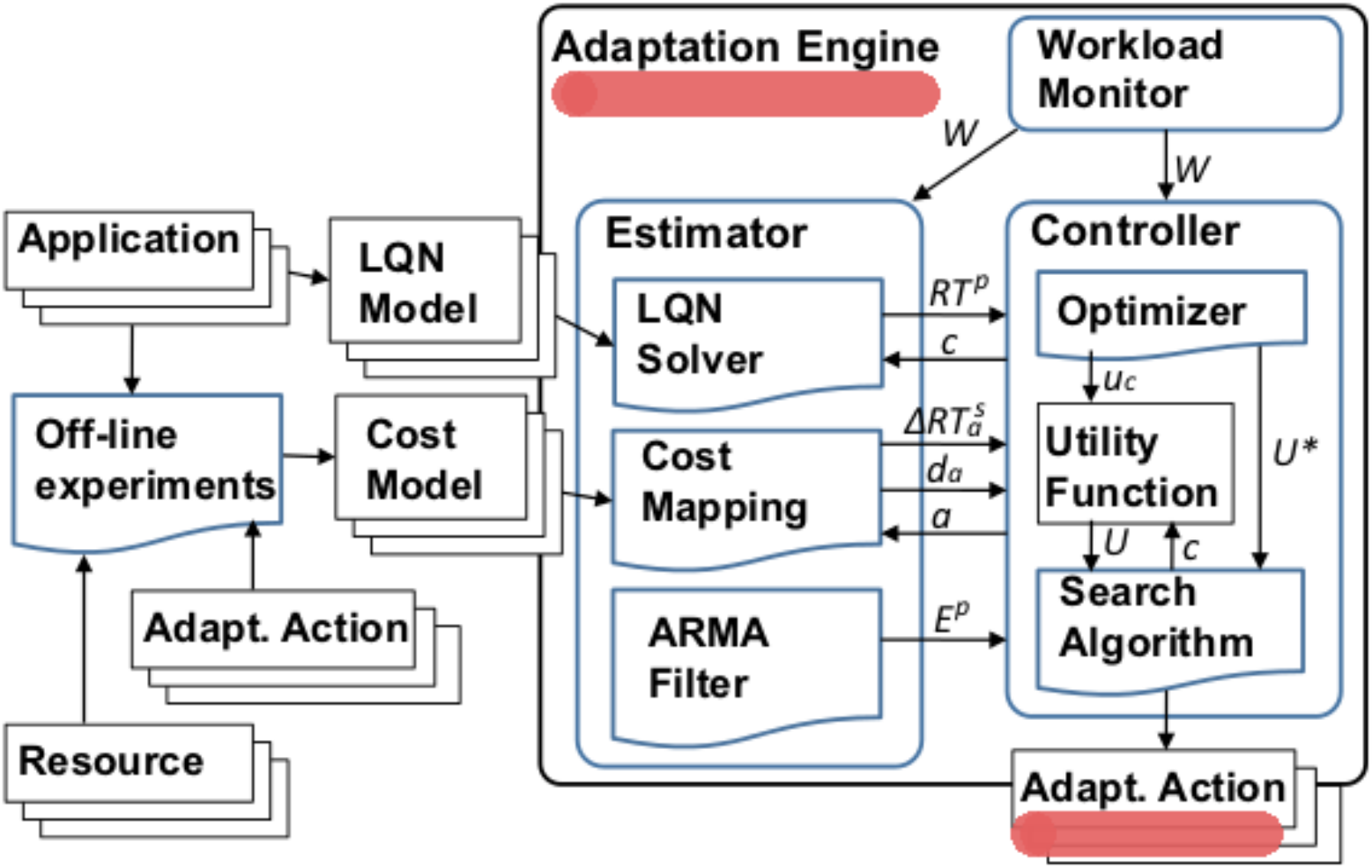
Impact of VM adaptations

- VM adaptations can be beneficial
- there ain't no such thing as a free lunch (TM)
- evaluate benefits taking costs into account

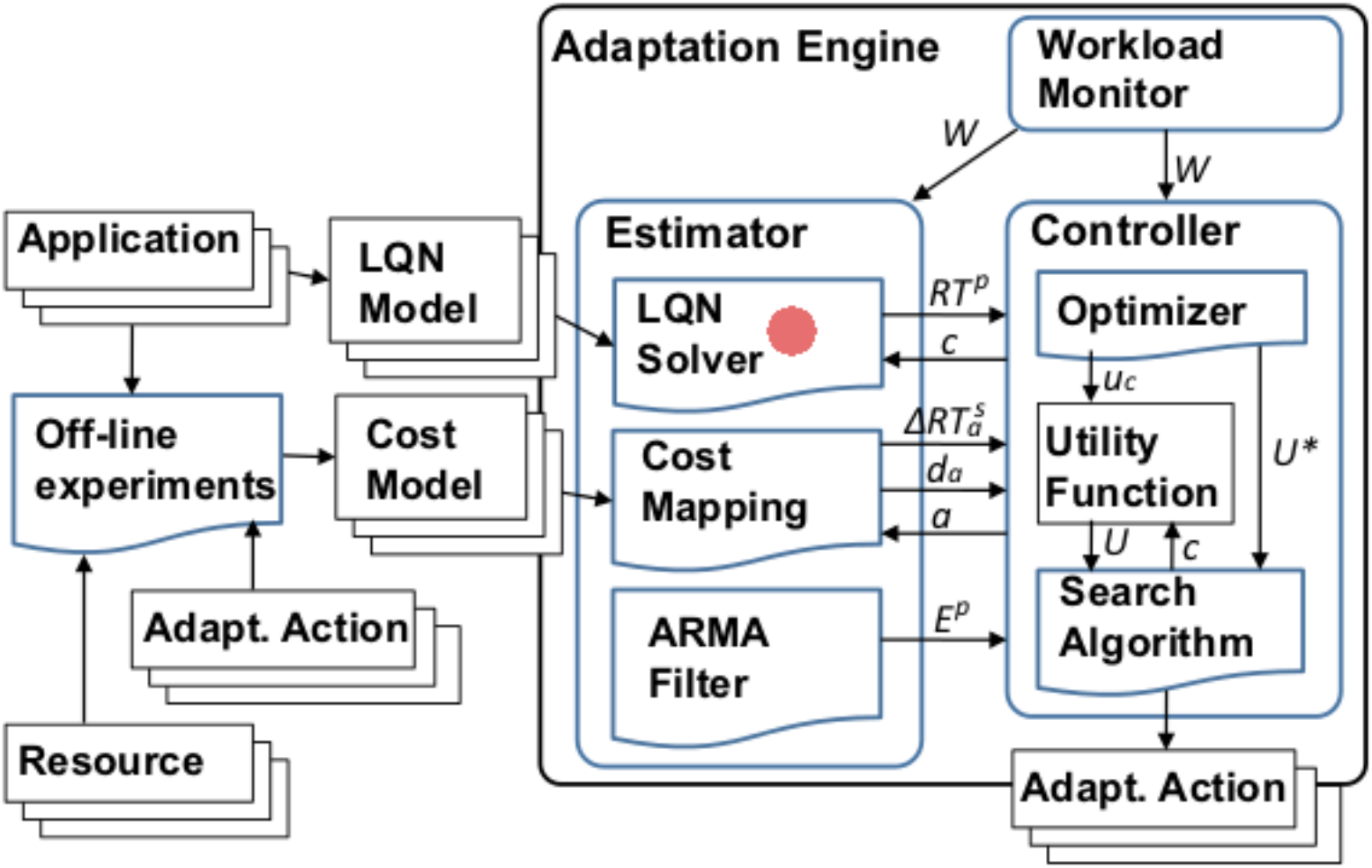
Cost-Sensitive Adaptation Engine



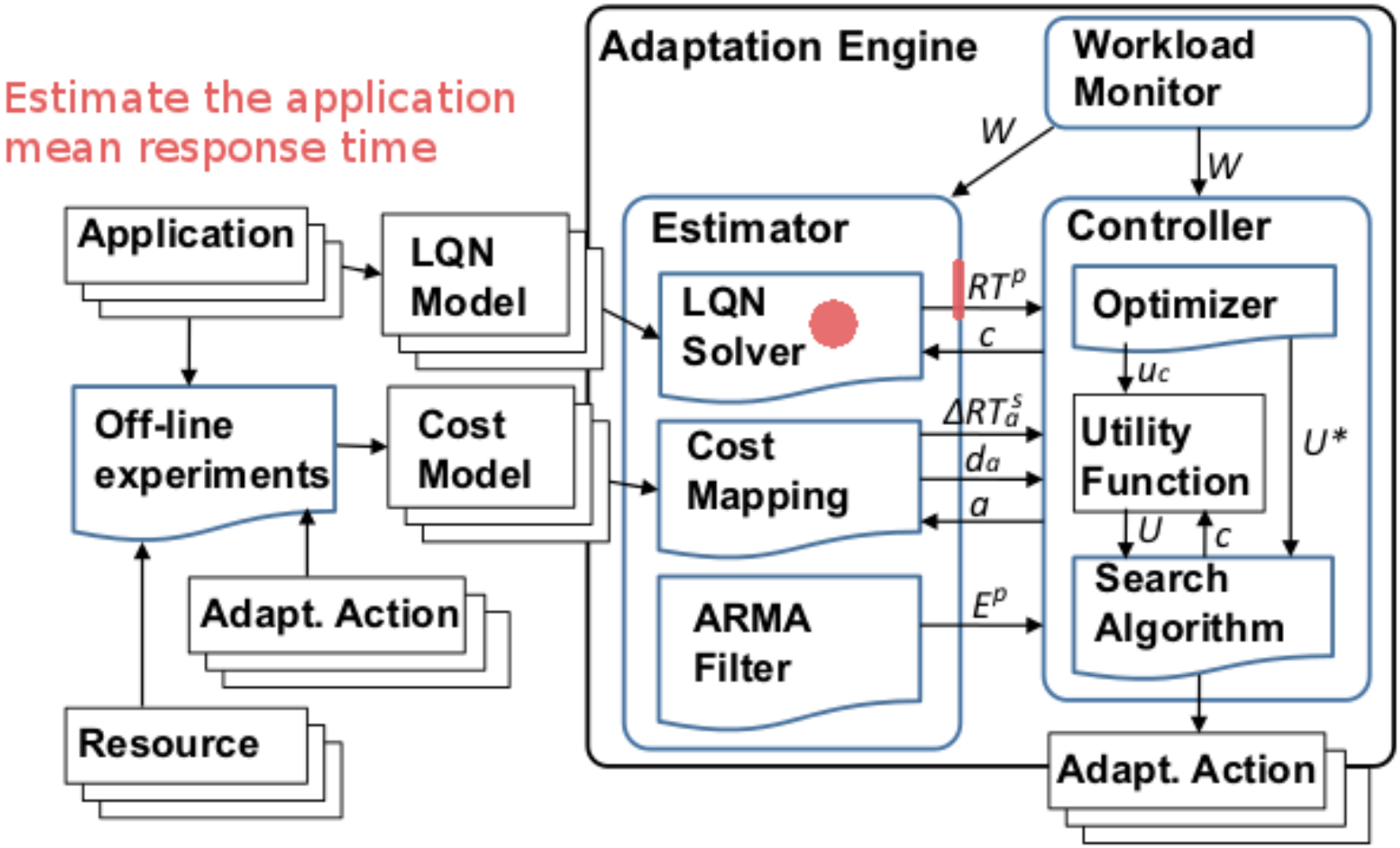
Cost-Sensitive Adaptation Engine



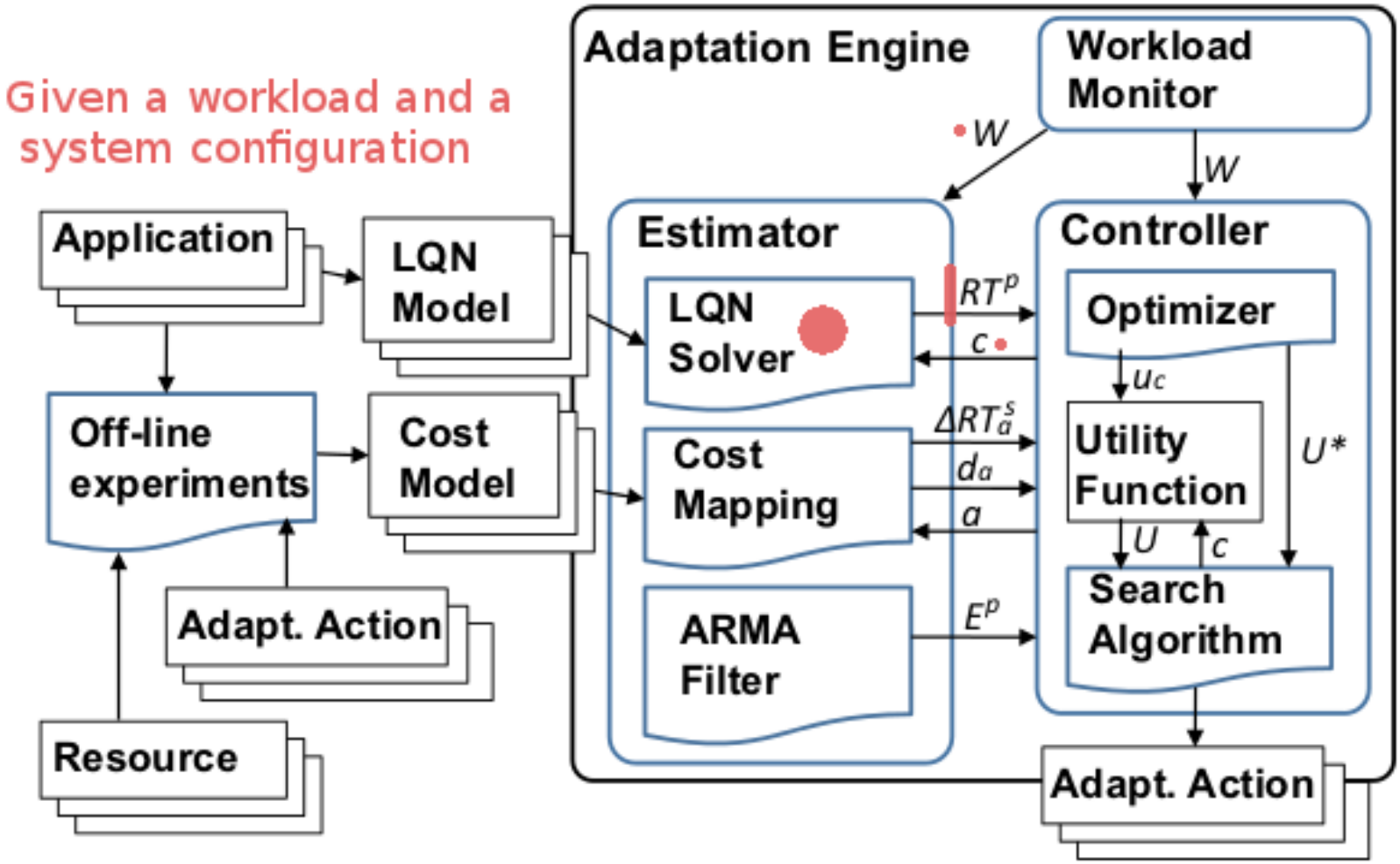
Cost-Sensitive Adaptation Engine: LQN Solver



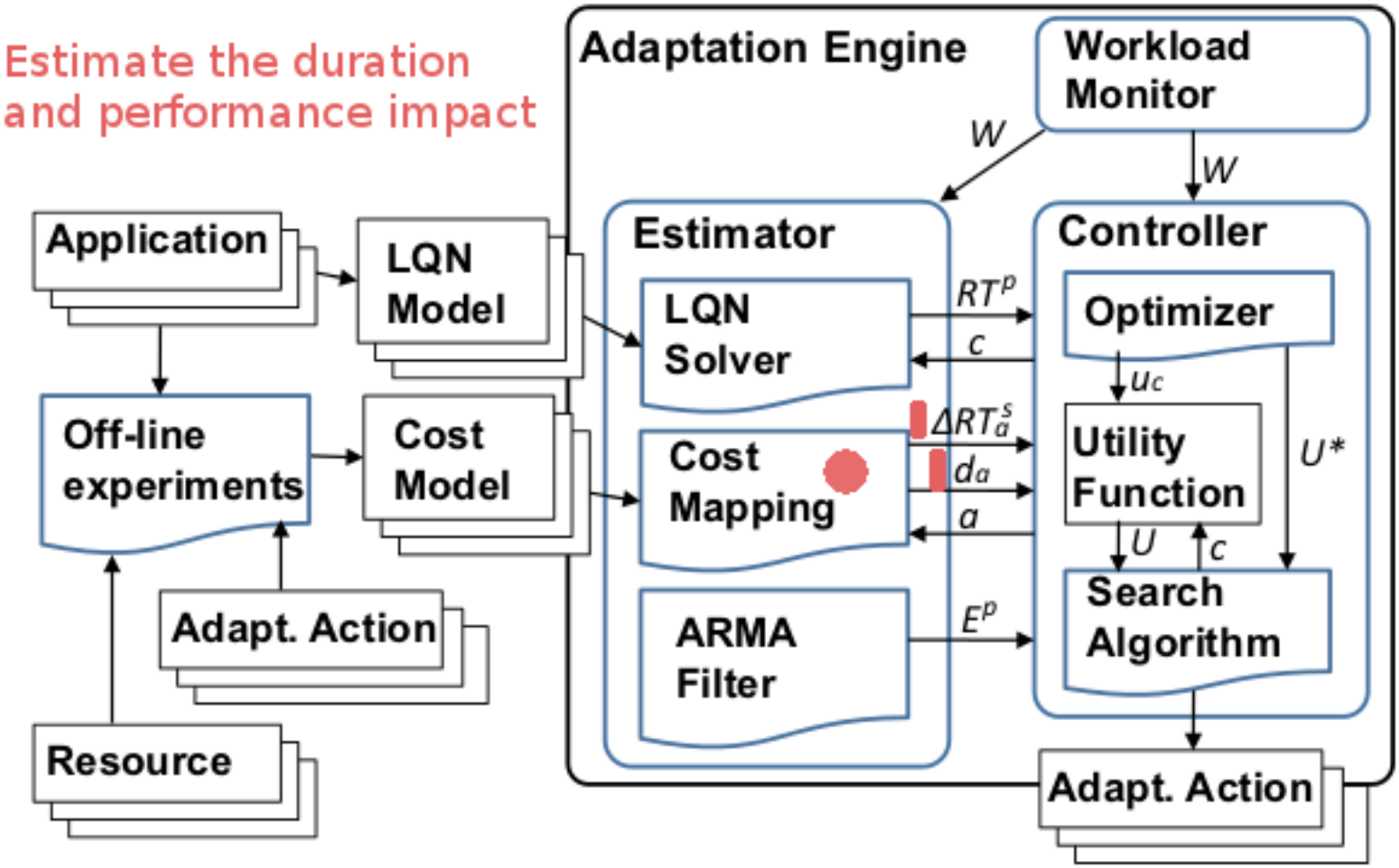
Cost-Sensitive Adaptation Engine: LQN Solver



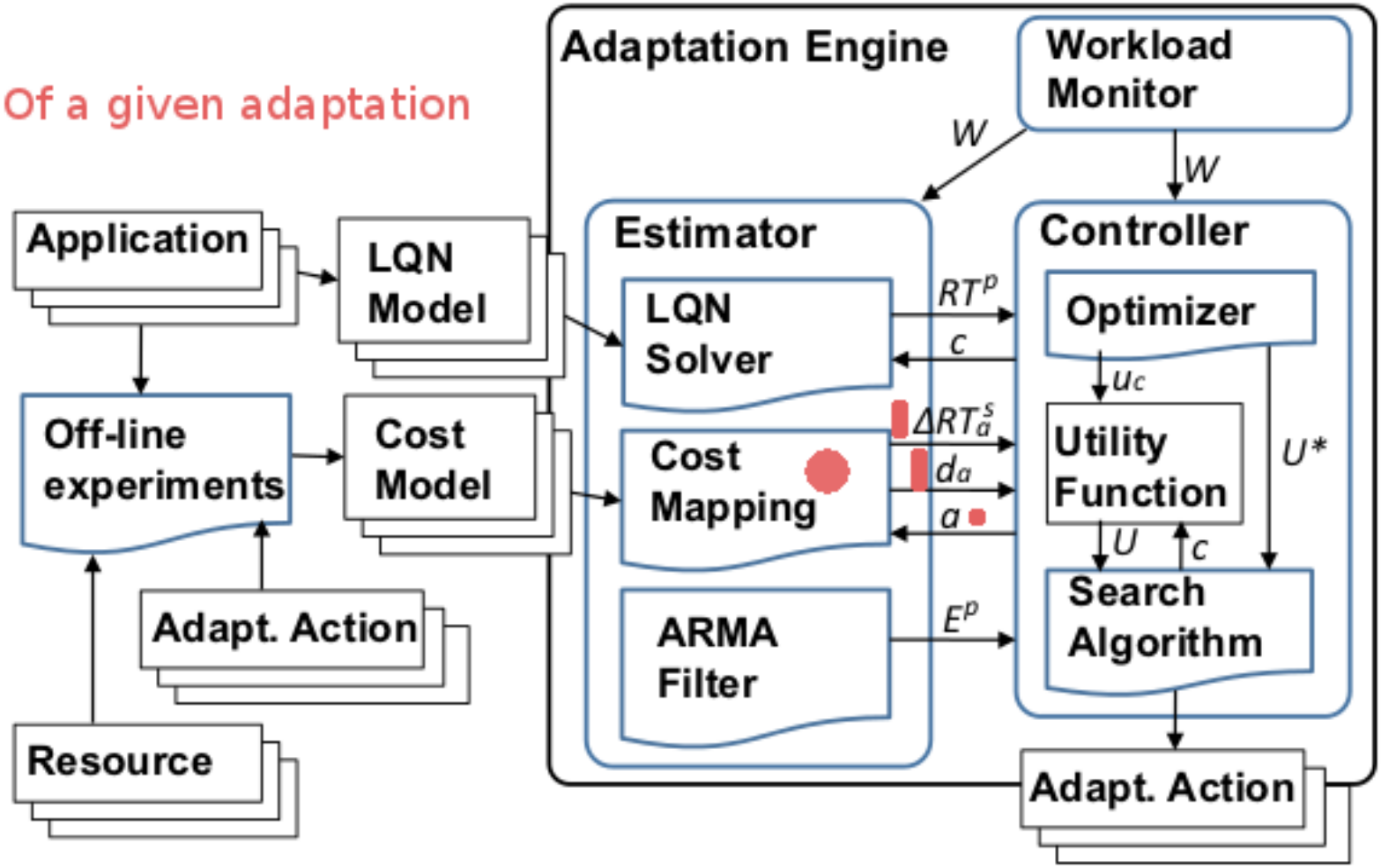
Cost-Sensitive Adaptation Engine: LQN Solver



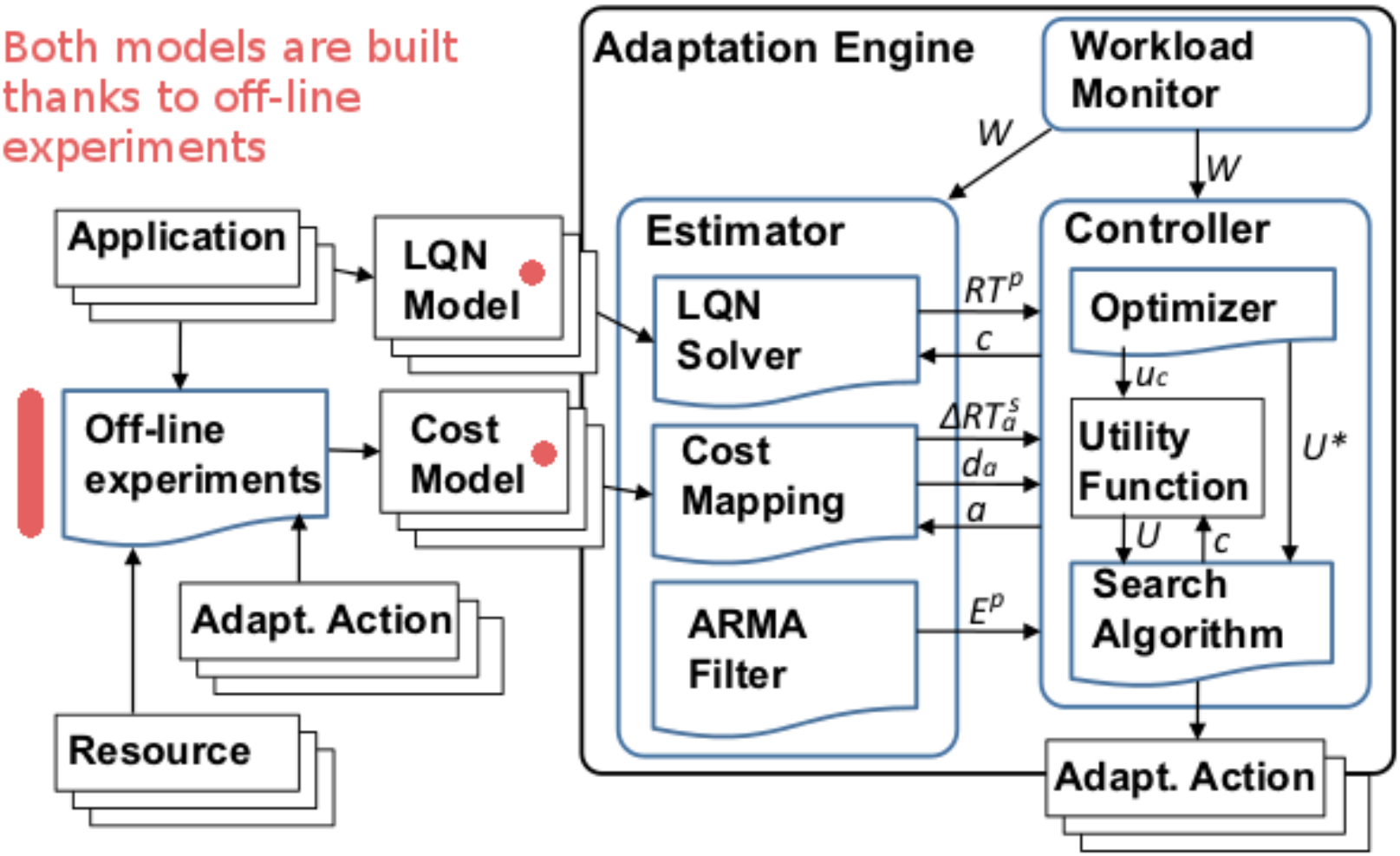
Cost-Sensitive Adaptation Engine: Cost Mapping



Cost-Sensitive Adaptation Engine: Cost Mapping

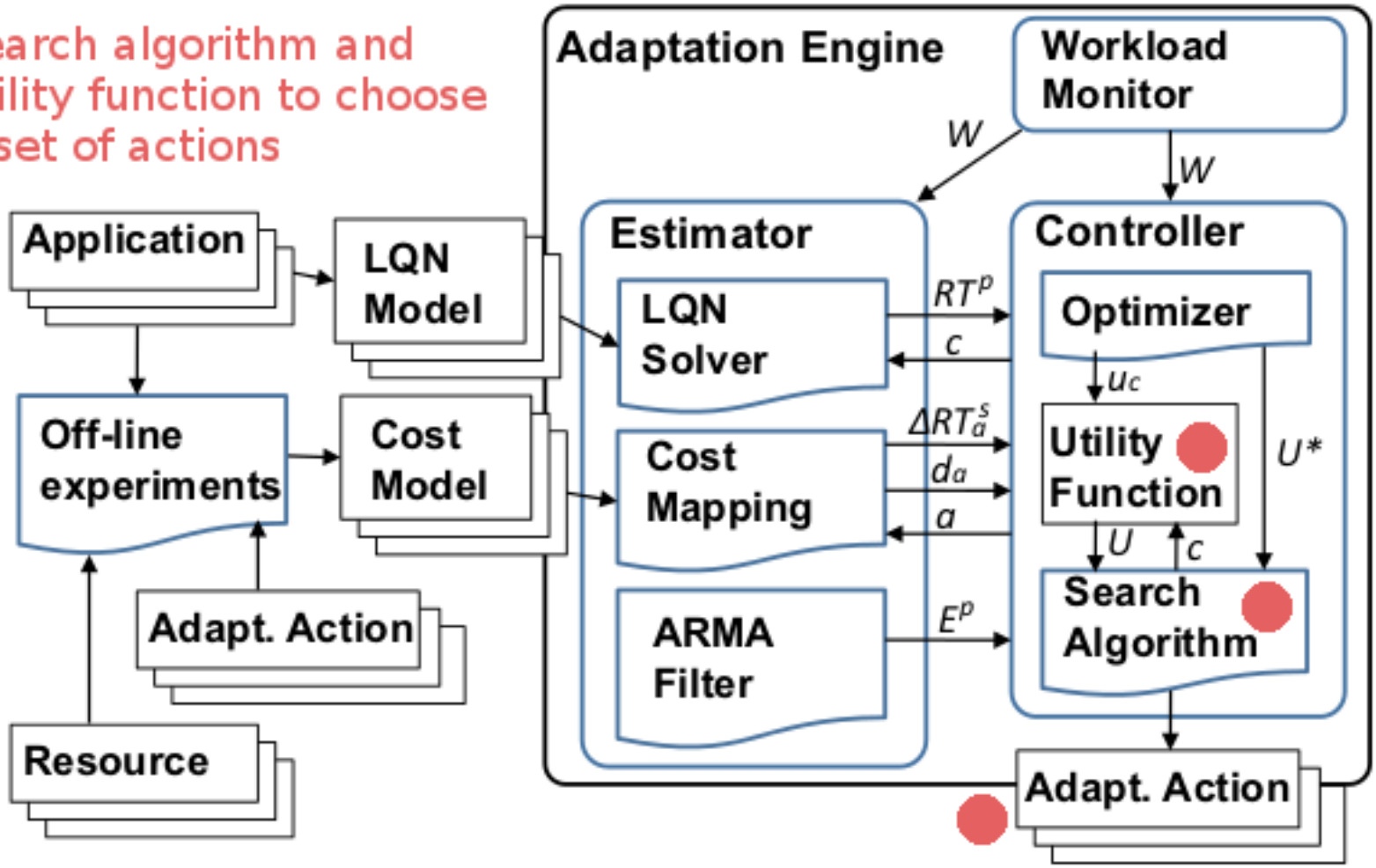


Cost-Sensitive Adaptation Engine: Off-line model parametrization



Cost-Sensitive Adaptation Engine: maximize utility

Search algorithm and utility function to choose a set of actions



Concerns

- Needs per-application offline experiments
- Tested on 4 machines, does it scale?
- World Cup request rates scaled from 1200 to 80 req/sec...

Conclusions

- Scalability can be seen from very different angles
- There is no "scale up" magic button
- We have seen how to design or refactor an application to improve its scalability
- How to provide a scalable hosting service fulfilling SLAs and without wasting money?