Challenges for cloud software engineering

Ian Sommerville

St Andrews University

Why is cloud software engineering different – or is it?

What needs to be done to make cloud software engineering easier for the research community?

Public, private community clouds

There will be a number of public cloud providers offering different capabilities at different prices at different times

Public cloud computing will be the most economical approach

Private (institutional) clouds for science research are not really viable

Community clouds make sense for data-intensive computing if they are located close to the sources of the data

Community clouds hosting standard software may be a useful approach – possibly hosted on public clouds

What makes the cloud different (and challenging)?

Scale

Thousands of nodes

Elasticity

No fixed hardware architecture

Scale out and scale in

Pricing

Variable costs for applications

Payment according to resources used

Cloud services

Infrastructure as a service

Instead of buying and running physical equipment (computers, storage, etc.), you 'rent' these as required from a cloud provider

You ship your programs and data to a remote data centre and run them there rather than locally

The familiar Grid computing model

Platform as a service

The cloud provider makes a set of APIs available which you use in your program to interact with the platform

Scaleability is transparent. Automatic scale out/scale in as demand changes

Primarily geared to writing web-based applications

Software as a service

Software is shared by different users and accessed using a web browser

Services are stateless, functional abstractions

Multi-tenant databases for efficient operation

Accessed via a service interface – SOAP or RESTful interface

Levels of cloud usage

1. Cloud hosting

You use the cloud to provide server capability rather than incurring the capital costs of server purchase

Applications need not be 'cloud aware'

Fixed server infrastructure

Can run existing codes in Fortran/C

If server type and platform available, then should not require any application changes

Almost certainly poorer performance than local execution.

But makes sense for occasional runs/ demonstrations etc.

Problems

Relatively low level model of interaction.

You need to know about lots of stuff (such as certificates, security keys, etc. that have nothing to do with your work)

Performance is difficult to predict

License issues

Costs may be high for high volumes of data transfer in/out of clouds

Software engineering challenge

Build a software development environment that radically simplifies hosting scientific applications on a range of clouds

2. Cloud interfacing

You build applications that make use of the cloud providers APIs (PaaS) to access common services.

'High-level' cloud awareness. The platform takes care of scaleability on demand.

Limited programming language support

Challenge is using existing PaaS APIs for computationally-intensive applications

- Systems are set up to support web-based applications
- Google Apps – 30 second time limit on tasks

Several experiments here – mixed conclusions.

Software engineering challenge

Investigate how to adapt applications that are computation/data intensive to run within the constraints set by the PaaS interfaces from cloud providers

3. Cloud software engineering

Build ‘cloud aware’ applications which can adapt to the cloud capabilities that are available

Application architecture designed for the cloud

Underlying infrastructure management done in the program itself

What abstractions do we need for this?

Three grand challenges

Programming models for the cloud

Building a PaaS for high performance/ throughput computing

Creating a cloud-aware software development environment

Scaling up and scaling out

Traditional model to cope with increased computational demand is to scale up

Supported by some cloud providers who offer different instance types.

Scaling out is the more usual cloud model

Coping with scaling out

Redesigning algorithms and code to work with a flexible rather than a fixed number of resources

- Interaction between program and platform to know what resources are being used and what resources are available
- Structuring program into a flexible number of computational components
- Avoiding sequential computations (Amdahl's Law)

Programming models

Are there programming models that can take advantage of elasticity and massive parallelism?

- Invent new models for parallel computation
- Adapt existing problems to the current map/reduce programming model

Map reduce explained

http://www.kchodorow.com/blog/2010/03/15/mapreduce-the-fanfiction/

Map-reduce on a slide

-- Split the data into a number of computationally tractable chunks.

```
in = split (thedata)
n = size (in)
```

-- **MAP.** For each chunk, process in parallel

```
parfor i = 1 to n do
{
    out (i) = F (in (i))
}
```

-- **REDUCE.** Apply a function R to the output that computes the required result
-- (of course, A can be a list)

```
A = R (out)
```

Map-reduce assumptions

Input data set can be split into chunks for independent parallel processing

The reduce action is a relatively simple, fast computation

Clearly, does not fit all types of problem

Can scientific codes be re-engineered to take advantage of the map-reduce model?

Elastic + static architecture

Use the elastic cloud for data preparation/analysis (MAP)

Use a cloud cluster for computation (REDUCE)

Opportunistic computing

Taking advantage of low-price/free resources when these are available?

Fixed budget computation

Compute till you run out of money

Get as much computational capability as your budget will allow

Use spot prices

Stop and start instances

Budget-driven programming

Develop systems that take advantage of variable pricing in different clouds/at different times

Potentially significant for computationally-intensive applications

PaaS for Science

Develop a set of APIS for data/compute intensive applications to support cloud interaction

- Long-running applications
- Cost/demand awareness
- Performance measurement/tuning
- Intermittent execution

SDE for the cloud

Creating an environment for the development of cloud-aware applications

Hiding low level details of IaaS cloud interaction

Testing environment

Occasional computation

Preliminary work – ELVIRA, St Andrews

http://www.elvira-cloud.org//

Software as a service

Effective way for a community to share services

Scope for applying external computations to collected data – data mining, visualisation, etc.

What are appropriate abstractions?

How can large amounts of state be managed?

Limiting data transfer between nodes

As with all software reuse, the community rather than the project benefits. Needs central funding to pay for reusable element creation

Conclusions

Clouds will become the dominant computing environment in future so costs will fall significantly – major driver for developing for the cloud

Offers a level of computational resource that cannot really be replicated in an institution

Software engineering issues need to be addressed to make the most effective use of the cloud