

Computing Resource Allocation and Management using Market Mechanisms

Date: 21 May 2008

Produced by: Chris Swan

The materials may not be used or relied upon in any way.

Agenda

A financial metaphor for the data centre

Where virtualisation fits into the picture

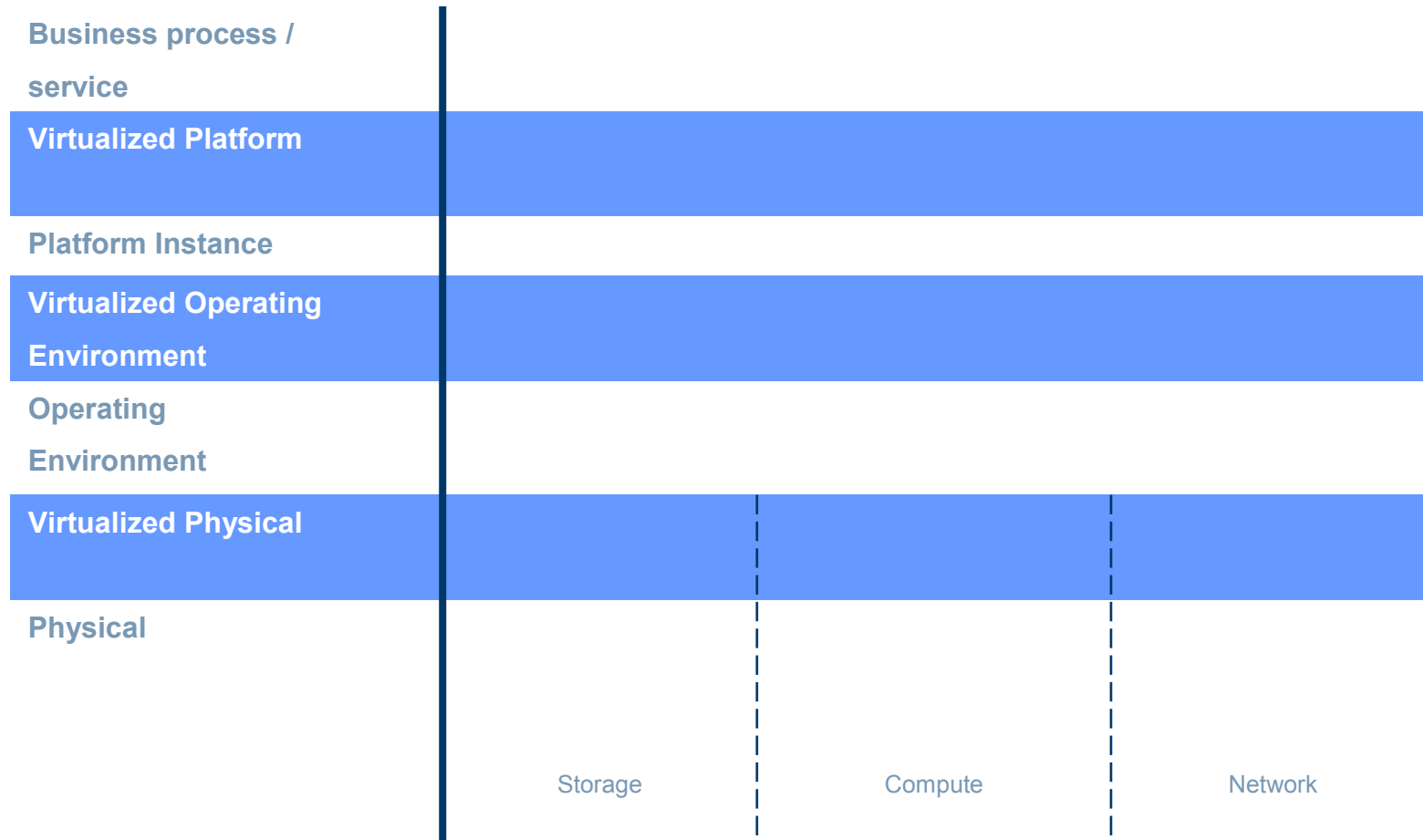
The virtual resource market

SLAs – the cornerstone to success

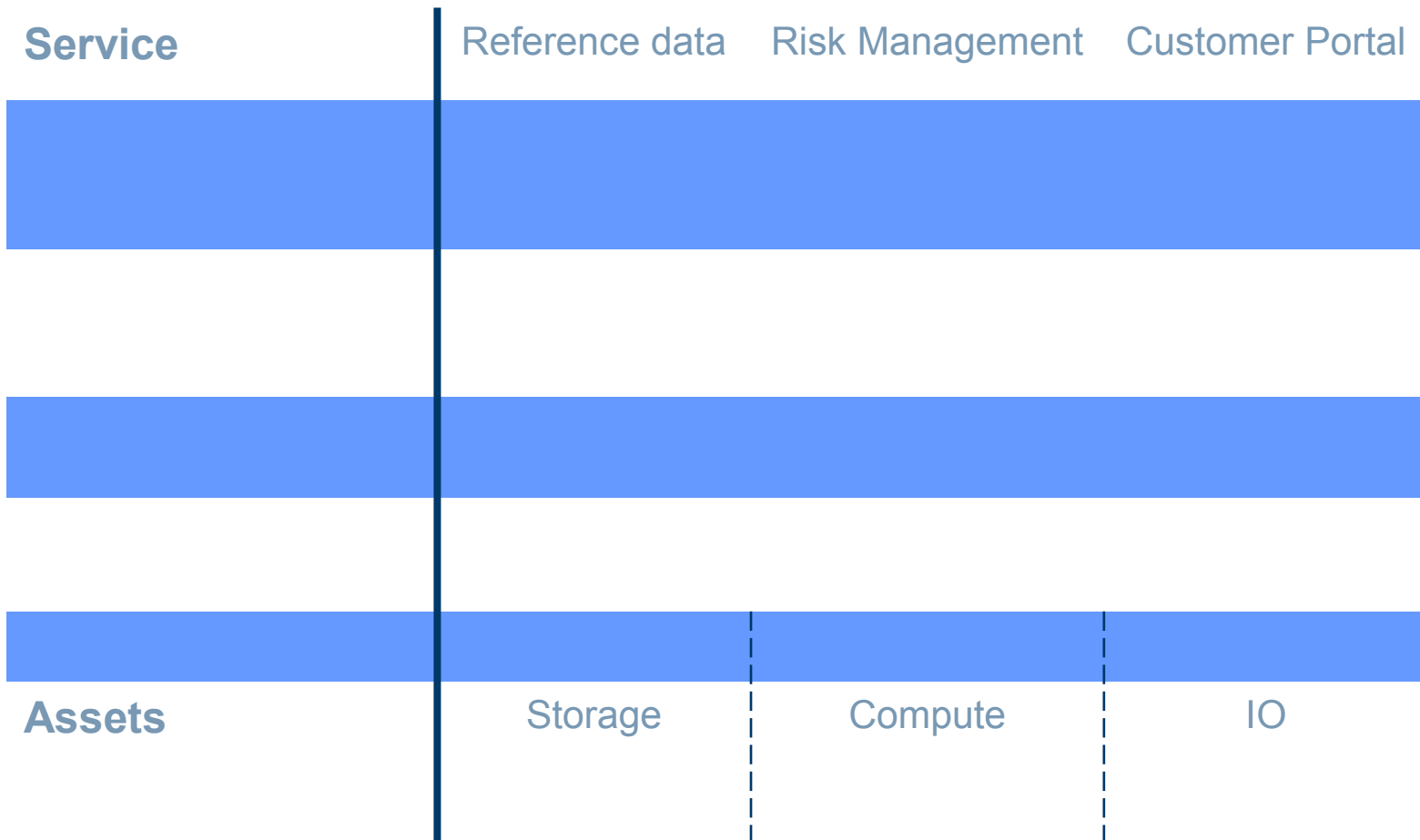
Integrating SLAs into the software development lifecycle

Questions

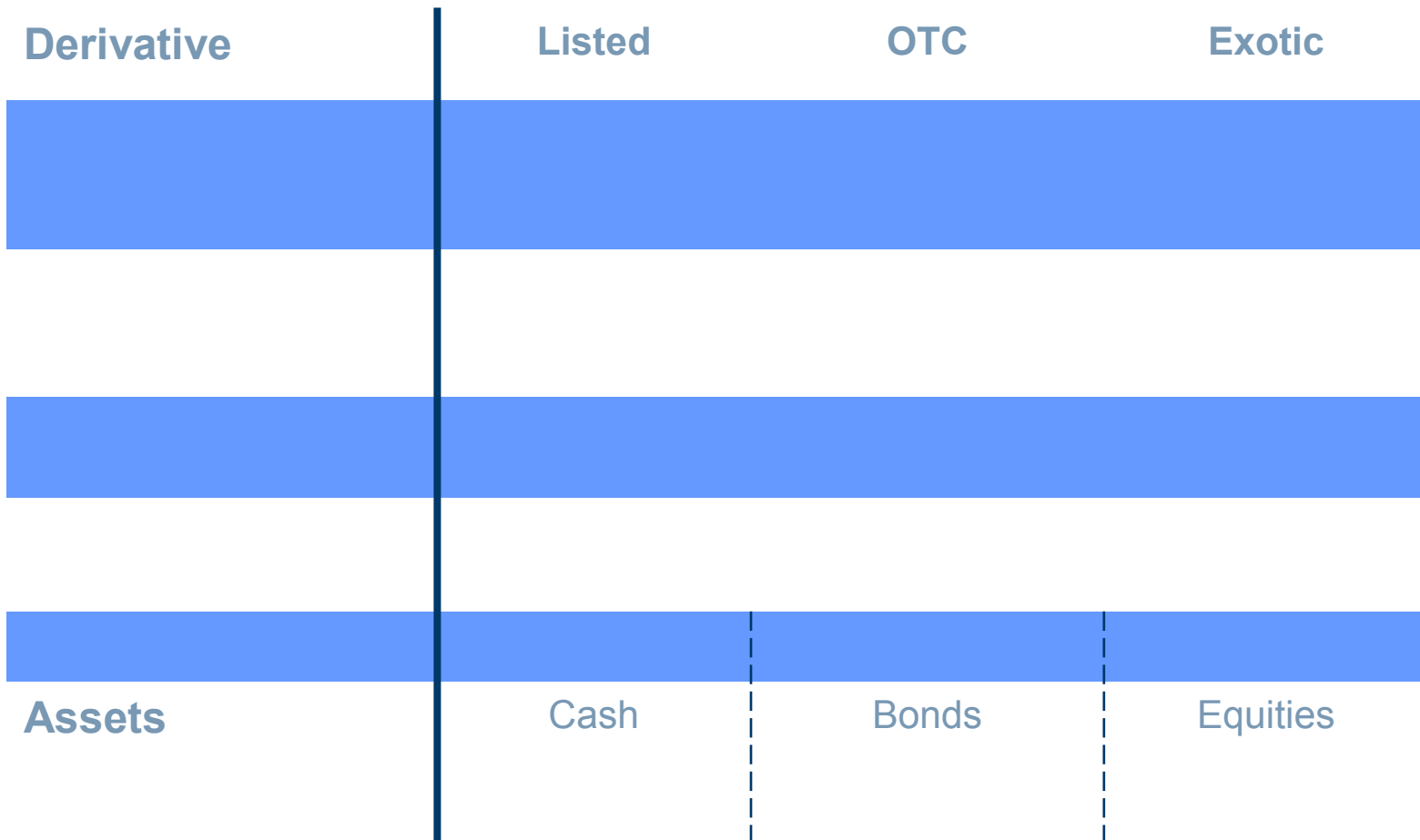
OGF technical reference model – axes only



OGF reference model – top and bottom layers only



OGF reference model - A financial metaphor



A layered view (from OGF technical reference model)

Business process / service	Reference Data	Risk Management	Customer Portal
Virtualized Platform	Data Grid	Compute Grid	Server Farm
Platform Instance	Database	App Server	Web Server
Virtualized Operating Environment	NFS, SMB, NAS	Virtual Machine Monitors	Load balancing, VIPs
Operating Environment	File systems e.g. NTFS, Ext3	Operating Systems e.g. Linux, Windows	Network protocols e.g. TCP/IP, UDP
Virtualized Physical	LUNs	Hypervisors	VLANs
Physical	Disks, Array Controller, SAN switches etc. Storage	Servers, Blades etc. Compute	Switches, Routers etc. Network

Each physical layer provides Abstraction to the layer above

Each Virtualized layer provides a flexible mapping/management point

Balancing the infrastructure

Service Level Agreements (SLAs)

Business process / service	Reference Data	Risk Management	Customer Portal
Virtualized Platform	Data Grid	Compute Grid	Server Farm
Platform Instance	Database	App Server	Web Server
Virtualized Operating Environment	NFS, SMB, NAS	VMs	Load balancing, VIPs
Operating Environment	File systems e.g. NTFS, Ext3	Operating Systems e.g. Linux, Windows	Network protocols e.g. TCP/IP, UDP
Virtualized Physical	LUNs	Hypervisors	VLANs
Physical	Disks, Array Controller, SAN switches etc. Storage	Servers, Blades etc. Compute	Switches, Routers etc. Network

Capacity & Performance

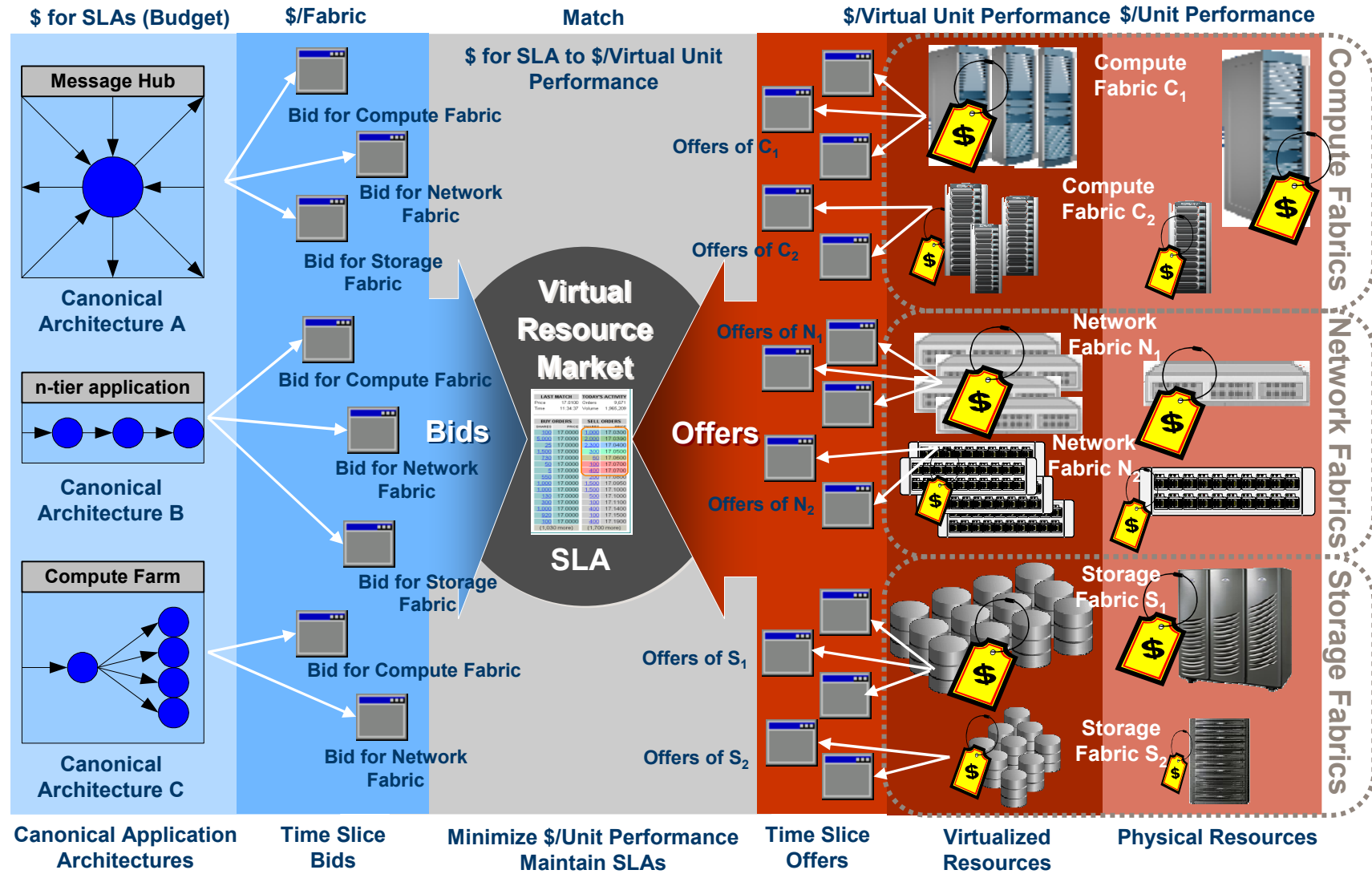


Management

(VRM)

Assets

Virtual Resource Market - Details



SLAs work just like any other piece of software

From the classic waterfall process (or SDLC+):

Initiation (Concept)

If we are going to have a system then we will need an SLA

Requirements definition

Identify at a coarse level what the parameters covered by the SLA will be

System and software design

Determine high level metrics (key performance indicators) then, refine to get specific metrics

Implementation and unit testing

This creates and verifies the functional parts of the SLA

Integration and system testing

At this stage it should be possible to validate that the non functional aspects are achievable

Deployment / maintenance

Ensure that the system performs within the SLA and responds to exceptions

Evaluation

Does the SLA actually represent the service to fit the business need that drove the original concept?



These stages are where efforts are typically focused with existing performance management tools.

Many systems are integrated and tested for 'ultimate' performance because no SLA has been defined, designed or developed earlier in the cycle.

Tools and technology

XML has become increasingly popular for modelling derivatives, with FPML emerging to cover most of the common ground

We need standard XSDs for SLAs

Composition is crucial – we don't code from scratch, so we won't build SLAs from scratch

Common models (canonical forms) can be reused

- These may well have repeatable behaviour as well as shape

Components and frameworks have yet to emerge

- SLAng (UCL) shows the way, WS-CDL may help with behaviour

Eclipse plugin for SLAs – coming soon?

Questions?

