Oxford's DAMS Architecture

Ben O'Steen
Software Engineer,
Oxford University Library Services

Neil Jefferies
R&D Project Manager
Oxford University Library Services

'An institutional repository needs to be a service with continuity behind it.....Institutions need to recognize that they are making commitments for the long term.'

Clifford Lynch, 2004

Continuity and Services

We anticipate that the content our systems will hold or even hold now will outlive any software, hardware or even people involved in its creation.

Continuity and Services

What content are we anticipating to store for the long-term?

Continuity and Services

- "Bookshelves for the 21st Century"
  - Legal Deposit
  - Voluntary Deposit
  - Digitisation
  - Extended Remit - Research Data
  - Administrative data – researchers, projects, places, dates
  - And anything else deemed worthy.

Continuity and Services

What core principles do you design a system with when you know the content will outlive you?

Continuity and Services

- **Oxford DAMS** – Digital Asset Management System
  - Not a single piece of software, more a set of guiding principles and aims.

Continuity and Services

#1 **Keep it simple**

– simple is maintainable, easy to understand.

Continuity and Services

- #1 **Keep it simple** – simple is maintainable, easy to understand.
  - Think about how much we have achieved with the simple concept of using a single command on a simple protocol (HTTP GET) to get an HTML document. (HTML being a loose, ill-constrained set of elements.)

Continuity and Services

- #1 **Keep it simple** – simple is maintainable, easy to understand.
  - simple protocols such as **HTTP**, simple formats such as **JSON** and API patterns such as **REST** are heavily favoured
  - Remember, at some point, administration and development of the system will need to be handed over.

Continuity and Services

#2 **Everything but the content is replaceable**

Continuity and Services

- #2 **Everything but the content is replaceable**
  - The content that is being held by the system at any one time is what is important, not the surrounding infrastructure
  - The content should survive and be reusable in the event that services like databases, search indexes, etc crash or corrupt.

Continuity and Services

1st logical outcome:

Do not use complex packages to hold your content!

Minimise the amount of software and work that you need to maintain to understand how your content is stored.

Continuity and Services

- 1st logical outcome:
  - Do not use complex packages to hold your content!
  - The basic, low-level digital object in our system is a “bag”
    - Each bag has a manifest, currently in either FOXML or RDF, which serves to identify the component parts and their inter-relationships.

Continuity and Services

#3 The services in the system are replaceable and can be rebuilt from the content.

Continuity and Services

- #3 The services in the system are replaceable and can be rebuilt from the content.
  - Services such as:
    - Search indexes (eg via Lucene/Solr)
    - Databases and RDF triplestores (MySQL, Mulgara, etc)
    - XML databases (eXist db, etc)
    - Object management middleware (Fedora)
    - Object registries

Continuity and Services

- 2nd logical outcome:
  - A content store must be able to tell services when items have changed, been created, or deleted, and what items an individual store holds.
  - We need slightly smarter storage to do this over a network. Local filesystems have had this paradigm for a long time now, but we need to migrate it to the network level.

Continuity and Services

- 2nd logical outcome:
  - A content store must be able to tell services when items have changed, been created, or deleted, and what items an individual store holds.
  - Each **store** and set of **services** needs a way to pass messages in a transactional way, a **messaging system**.
  - [Tech note: we are moving from the JMS to use the new AMQP via Rabbit MQ because on balance it is much more practical, flexible and useful.]

Continuity and Services

- 3rd logical outcome:
  - The storage must store the information and documentation for the standards and conventions its content follows.
  - The storage must be able to describe itself.

Continuity and Services

#4 We must lower our dependence on any one node in the system – be it hardware, software or even a person.

Continuity and Services

- #4 We must lower our dependence on any one node in the system – be it hardware, software or even a person.
  - This is another reason for #1 – keeping things simple.
    - Lowers the cost of hardware replacement and upgrade – simple hardware tends to be cheaper and more readily available; lowering the number of crucial features broadens what can be used.
    - Software – Oxford has been around for 1000 years. Vendors will not be.
    - People – a simpler system is more readily understood by new/replacement engineers.

Challenges to maintaining Continuity

- Flexibility
- Scalability
- Longevity
- Availability
- Sustainability
- Interoperability

Challenges to maintaining Continuity

- Flexibility
  - Open standards and open APIs allow us to provide the tools people need to create the experience with the content they want.
  - 'Text' based metadata – XML, RDF, RDFa, JSON
  - Componentised web services providing open APIs – flexibility through dynamically selecting what services run at anytime.
  - Open Source

Challenges to maintaining Continuity

- Scalability
  - We need to anticipate exponential growth in demand, with the knowledge that storage will be longterm
  - A rough estimate of all the data produced in the world every second is 7km of CDROMs – broadcast video, big science, CCTVs, cameras, research, etc. This figure is only ever going to go up.
  - To scale like the web, we have to be like the web; not one single black box and workflow, but a distributed net of storage and services, simply interlinked.
  - The “Billion file [inode] problem” must be avoided

Challenges to maintaining Continuity

- Longevity
  - Live Replicas (Backup Problems)
    - MAID (Power)
  - Self-healing Systems (Resilience)
  - Simplicity of Interfaces
  - Avoid 3rd Party dependence
  - Support Heterogeneity
    - Resolvers/Abstraction Layers

Challenges to maintaining Continuity

- Availability
  - Basic IT availability
  - Enhanced long-term availability
  - Archival recoverability
  - Digital Preservation
    - Conversion
    - Emulation
    - Archive Preservation

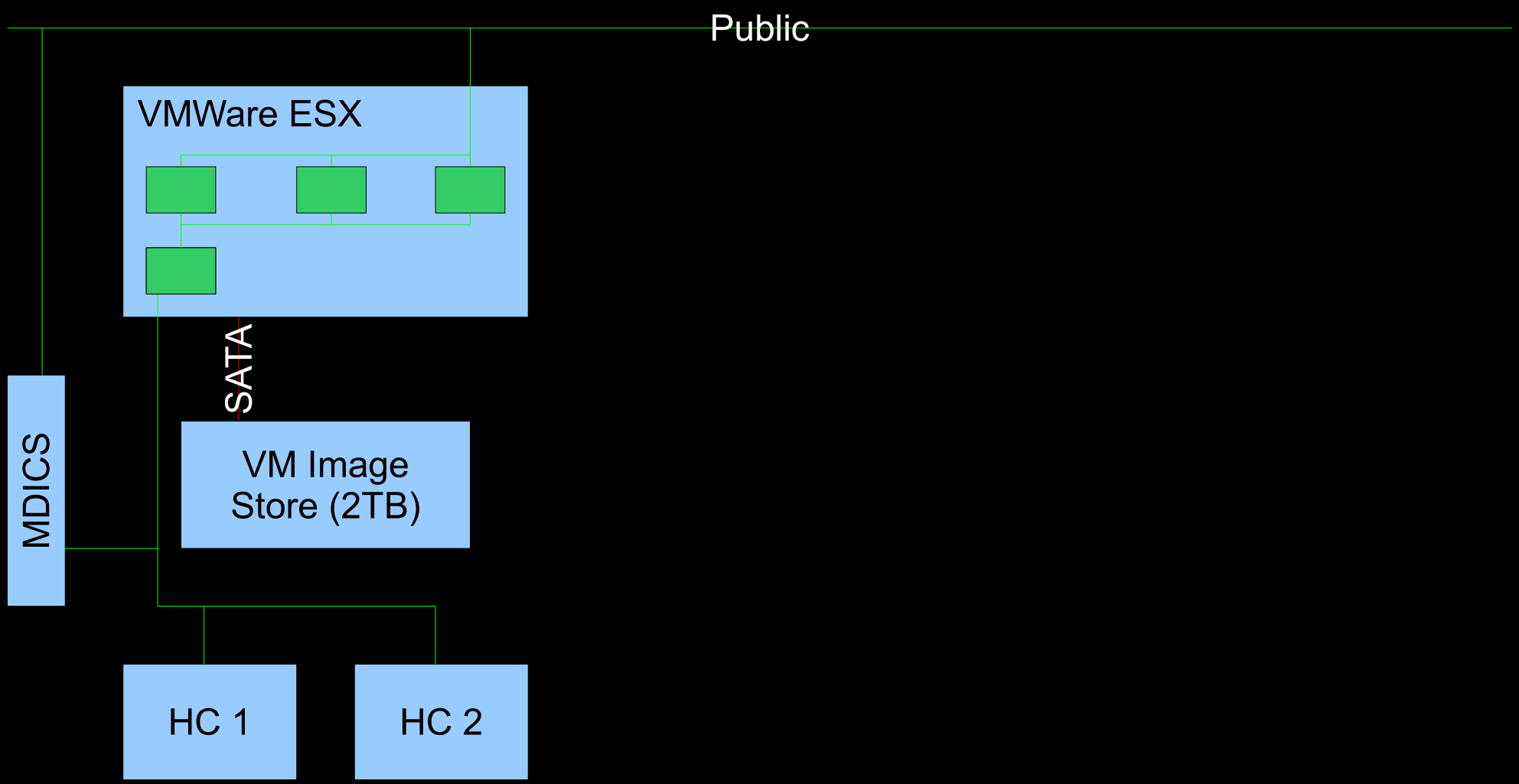
Challenges to maintaining Continuity

- Sustainability
  - Budget and cost as a **conventional** library
  - Factor archival costs into projects
  - Leverage content to generate income
  - Migrate skills

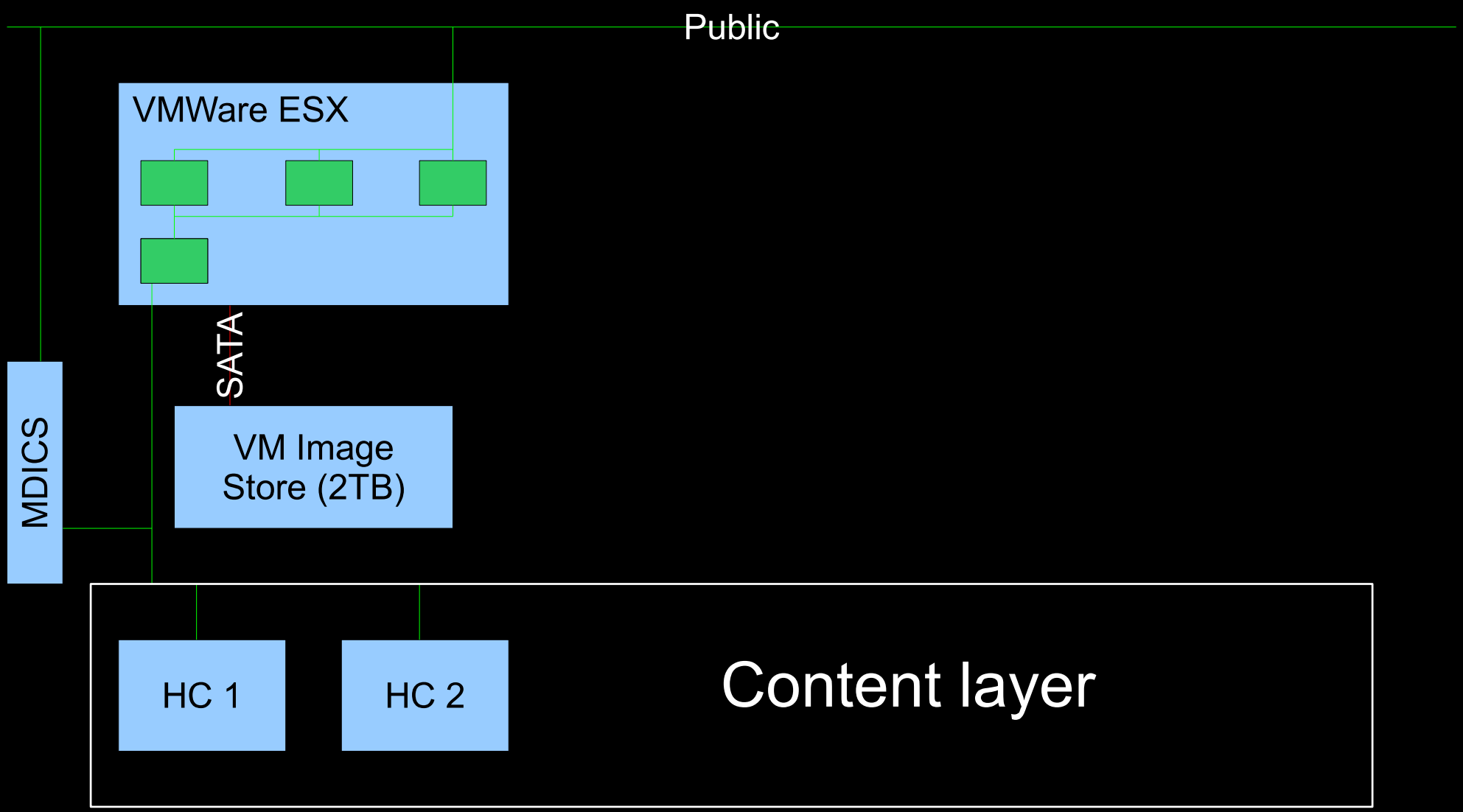
Challenges to maintaining Continuity

- Interoperability
  - Interoperability is an ongoing process
    - Support for emerging and established standards
  - Persistent, stable, well-defined interfaces
  - Ideally implement interfaces bidirectionally
  - Avoid low-level interfaces – abstract as much as feasible
  - Embrace the web – if you do it right, people will reuse your content in ways you never thought of and never thought possible.

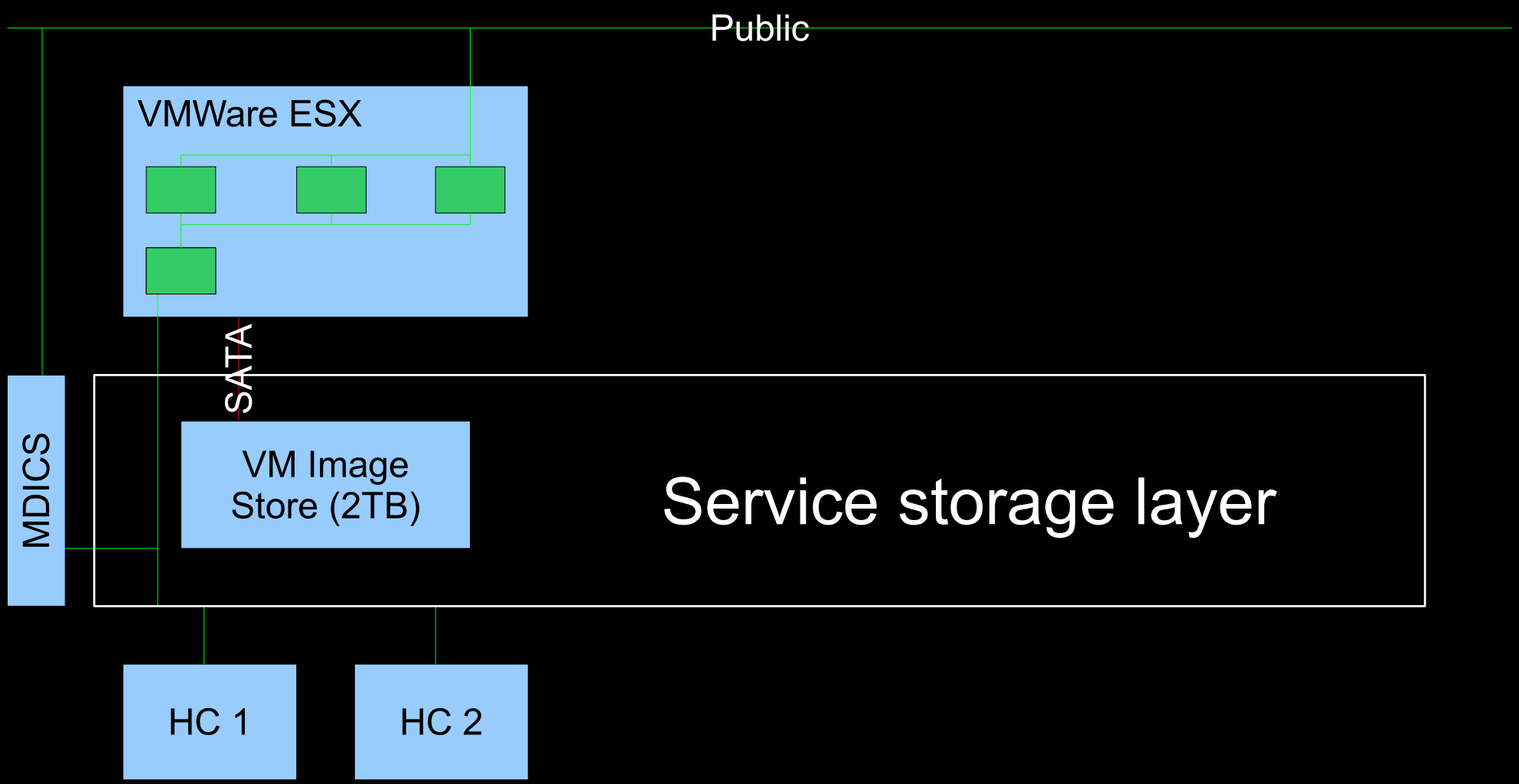
Phase 1 – current hardware



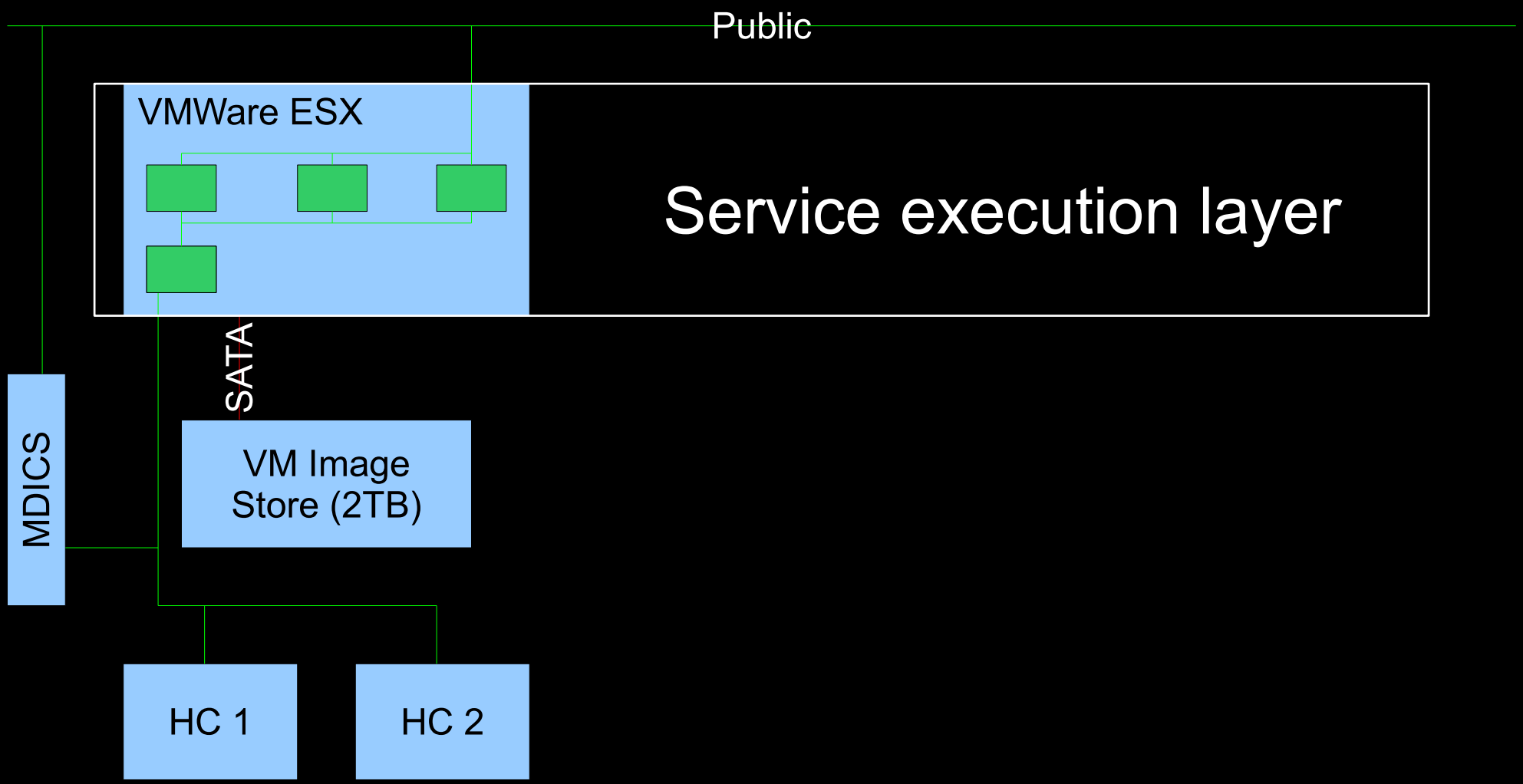
Phase 1 – current hardware



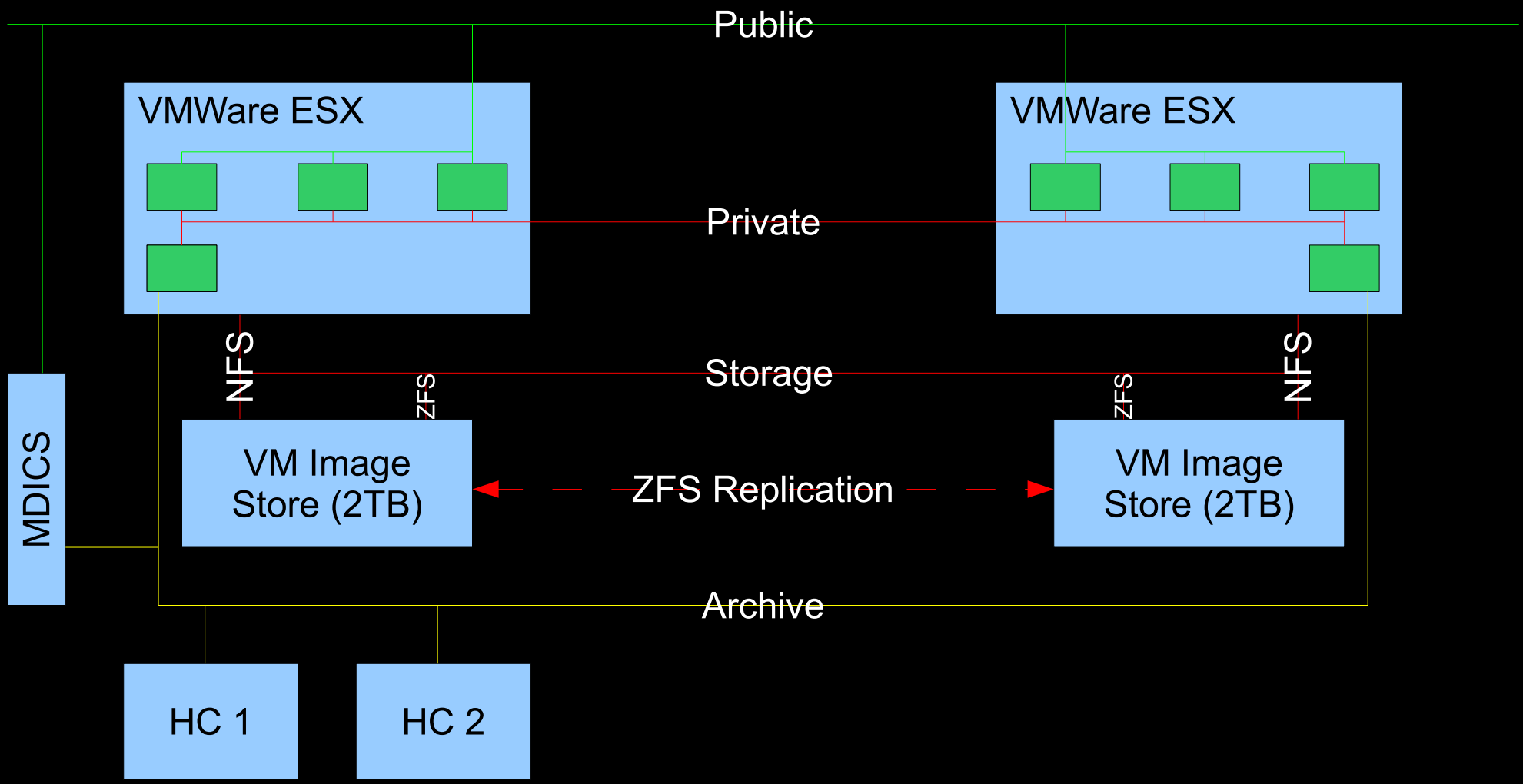
Phase 1 – current hardware



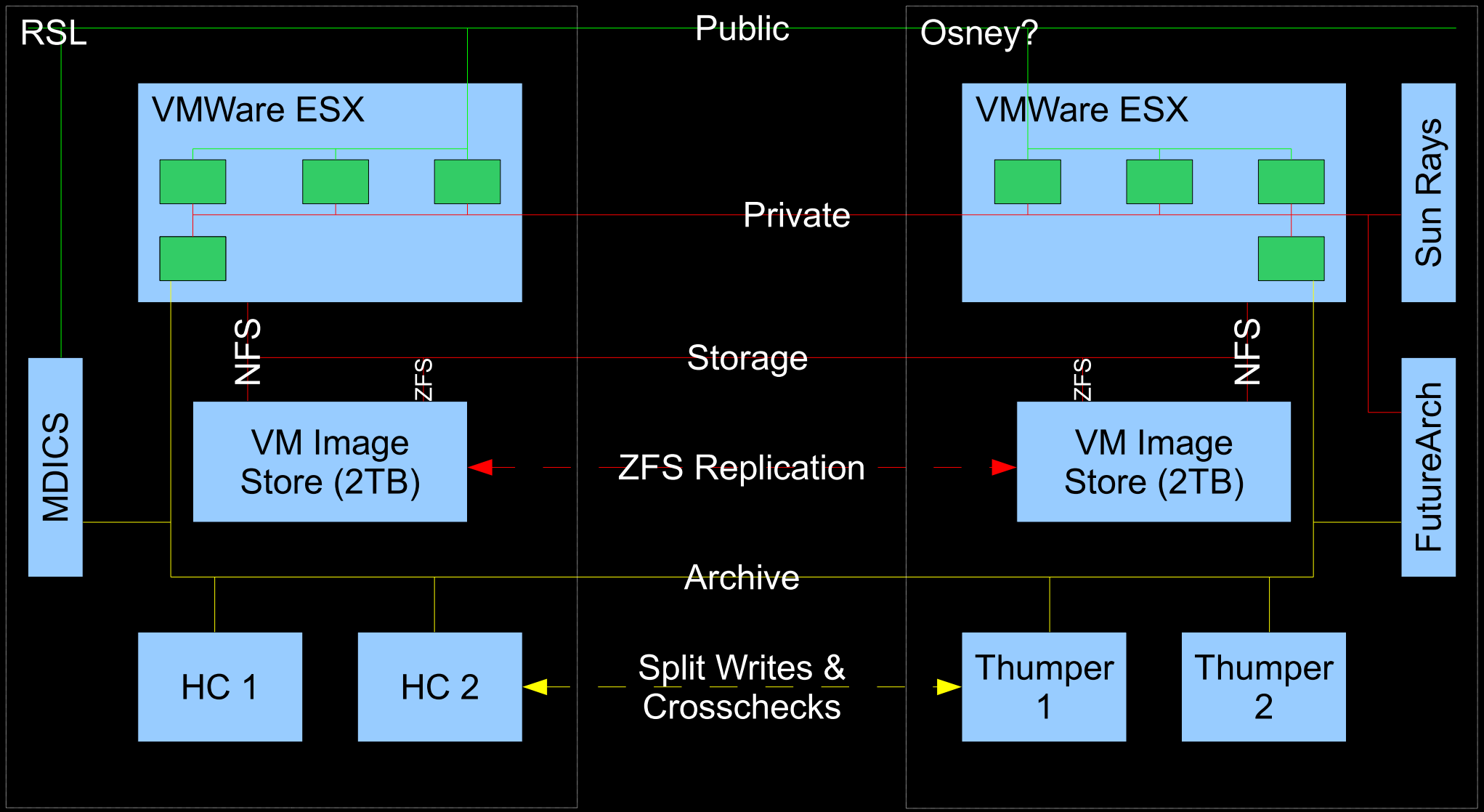
Phase 1 – current hardware



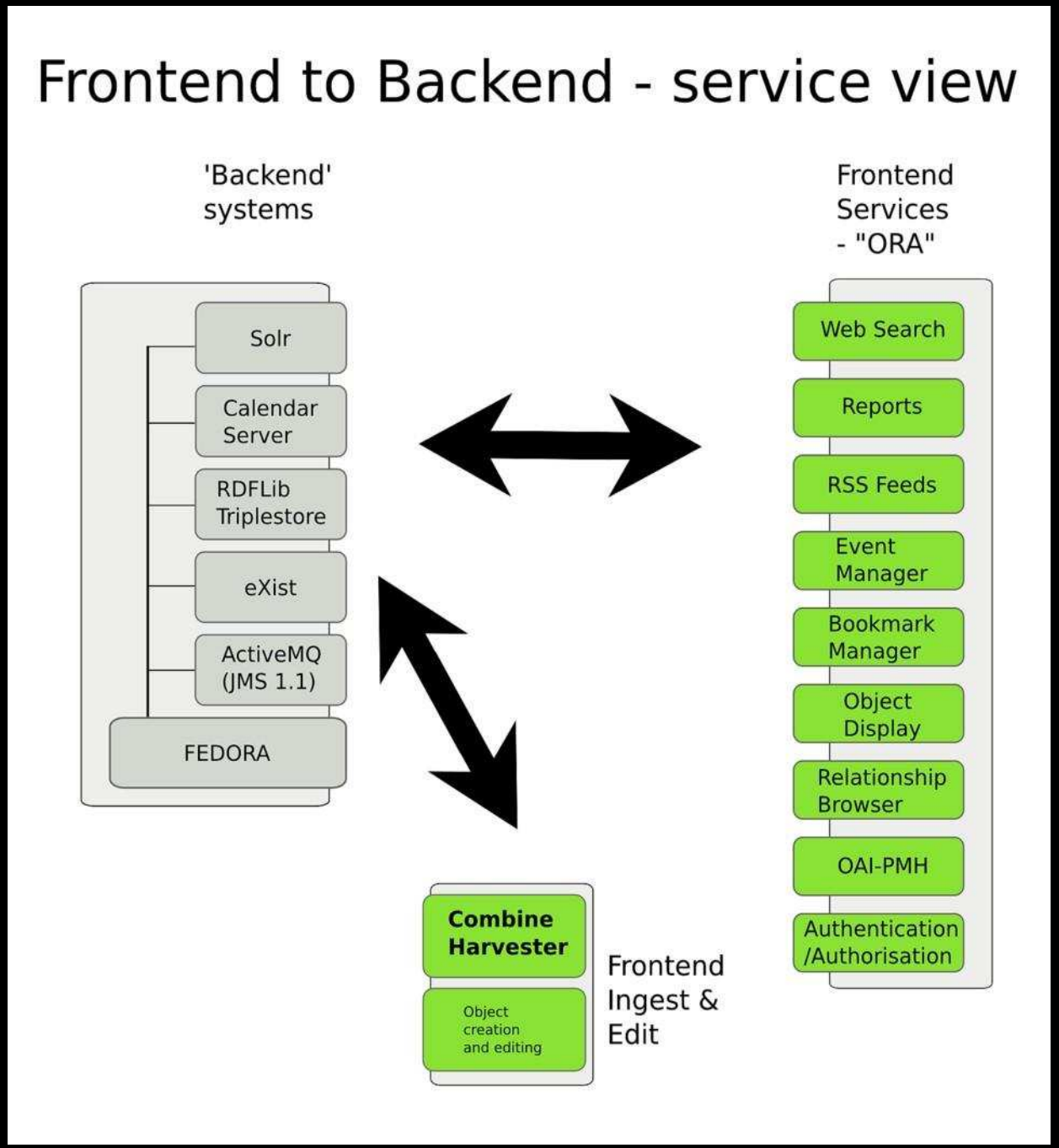
Phase 2 – (expecting hardware delivery this month)



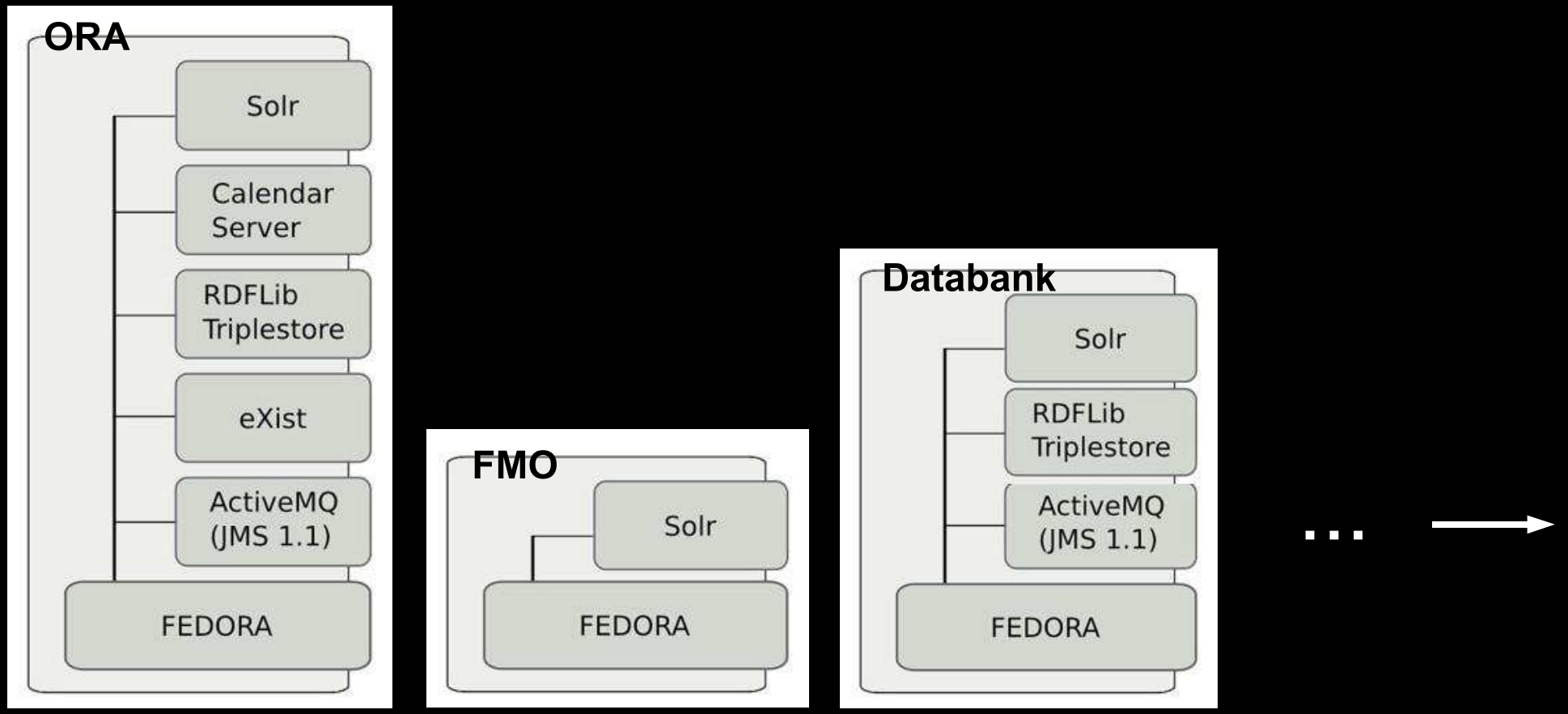
Aim – multisite (add sites as required to scale up)



Backend services (blue) fit over the content storage in 'stacks' and provide capability to other services (green)



Columns of services over common content/storage layer



Storage layer

End of Presentation

Any Questions?

Ben O'Steen
benjamin.osteen@ouls.ox.ac.uk

Neil Jefferies
neil.jefferies@sers.ox.ac.uk
www.sers.ox.ac.uk

Oxford Research Archive
http://ora.ouls.ox.ac.uk

Developer's Blog
http://oxfordrepo.blogspot.com