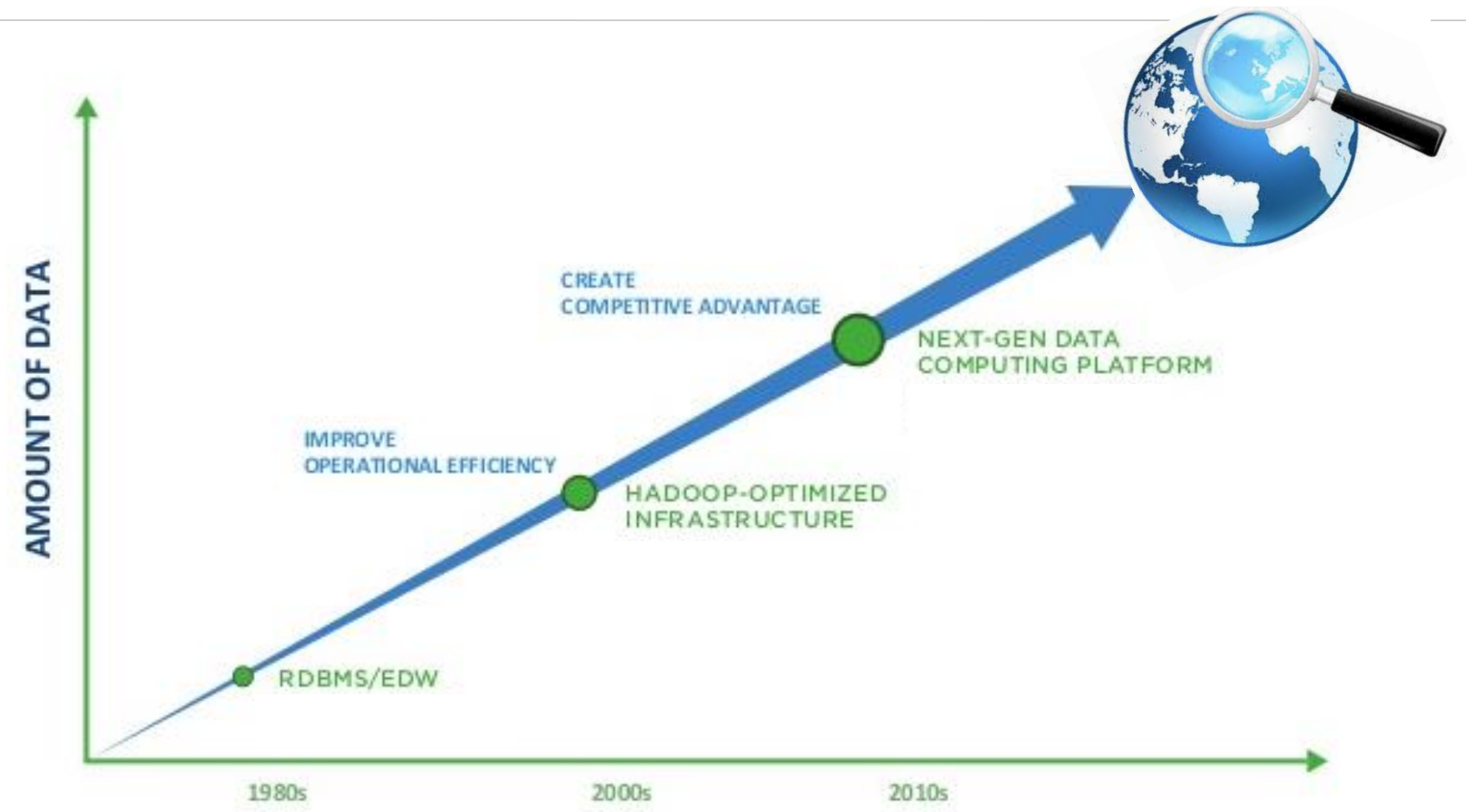
Big Data Fuels IT Architecture Evolution

@EvaAndreasson, Cloudera

Data Re-Thinking Drivers

Where we are Heading…

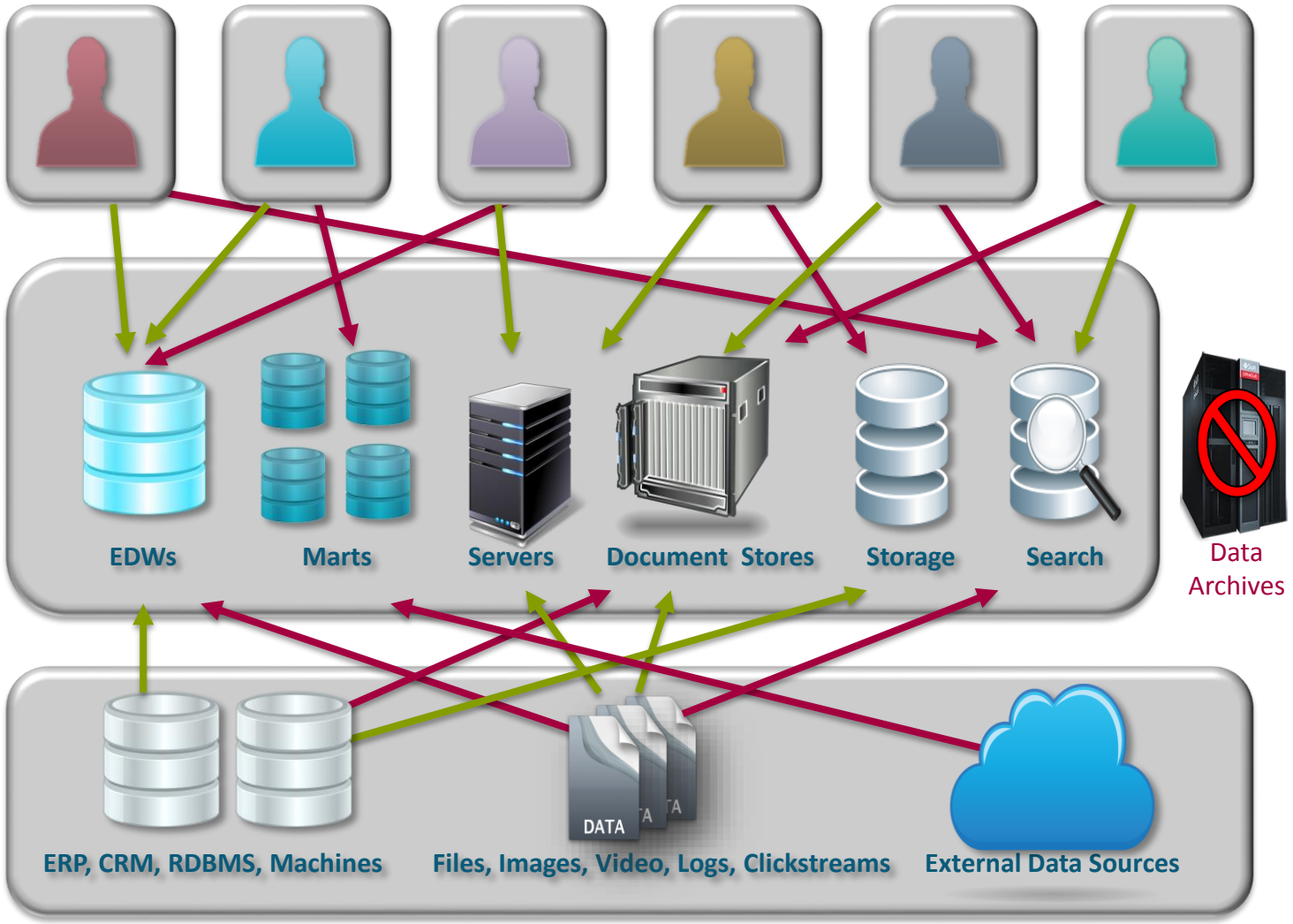


The Need to Rethink Data Architecture

Thousands of Employees & Lots of Inaccessible Information

Heterogeneous Legacy IT Infrastructure

Silos of Multi-Structured Data Difficult to Integrate

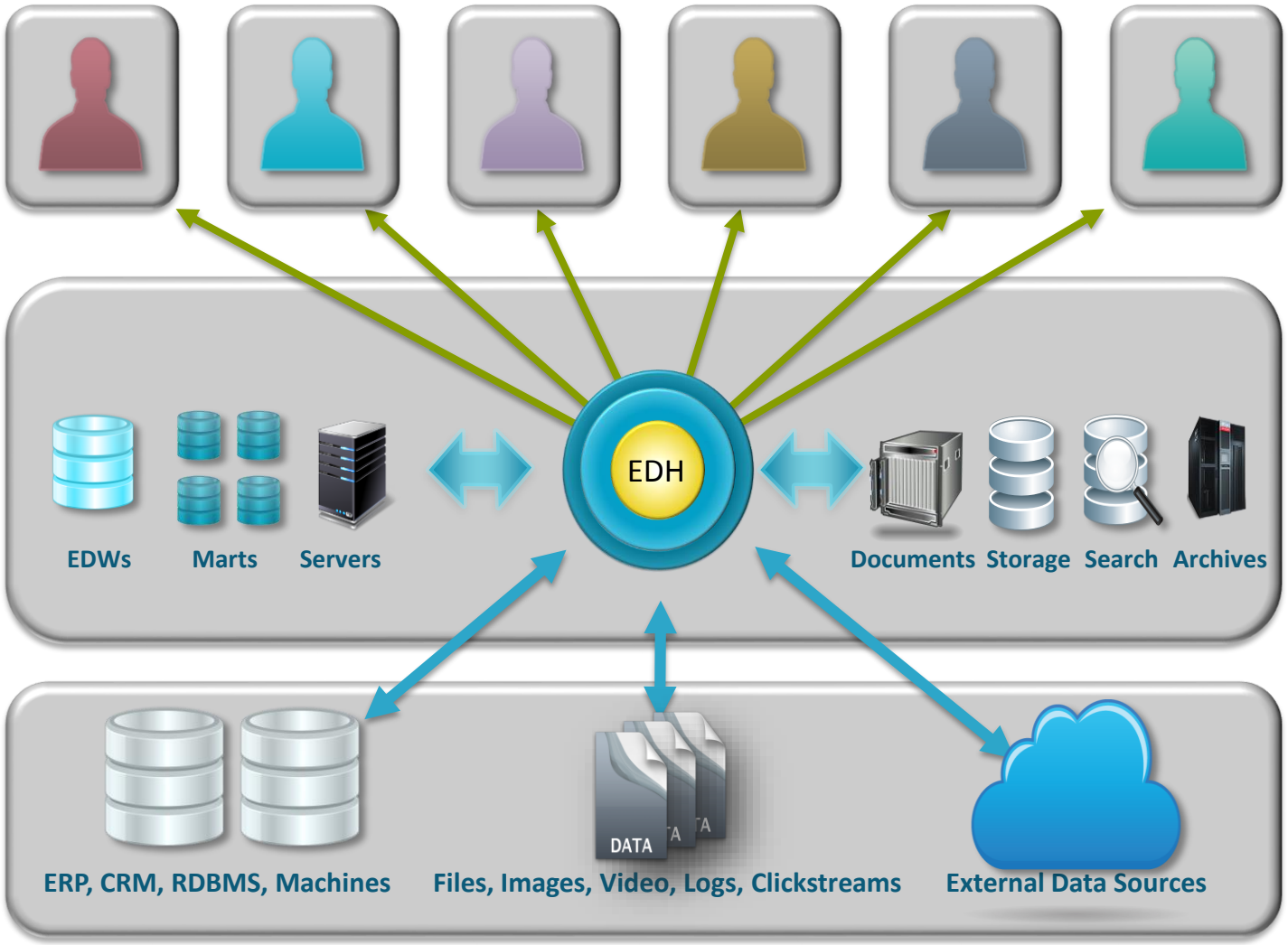


New Category: The Enterprise Data Hub (EDH)

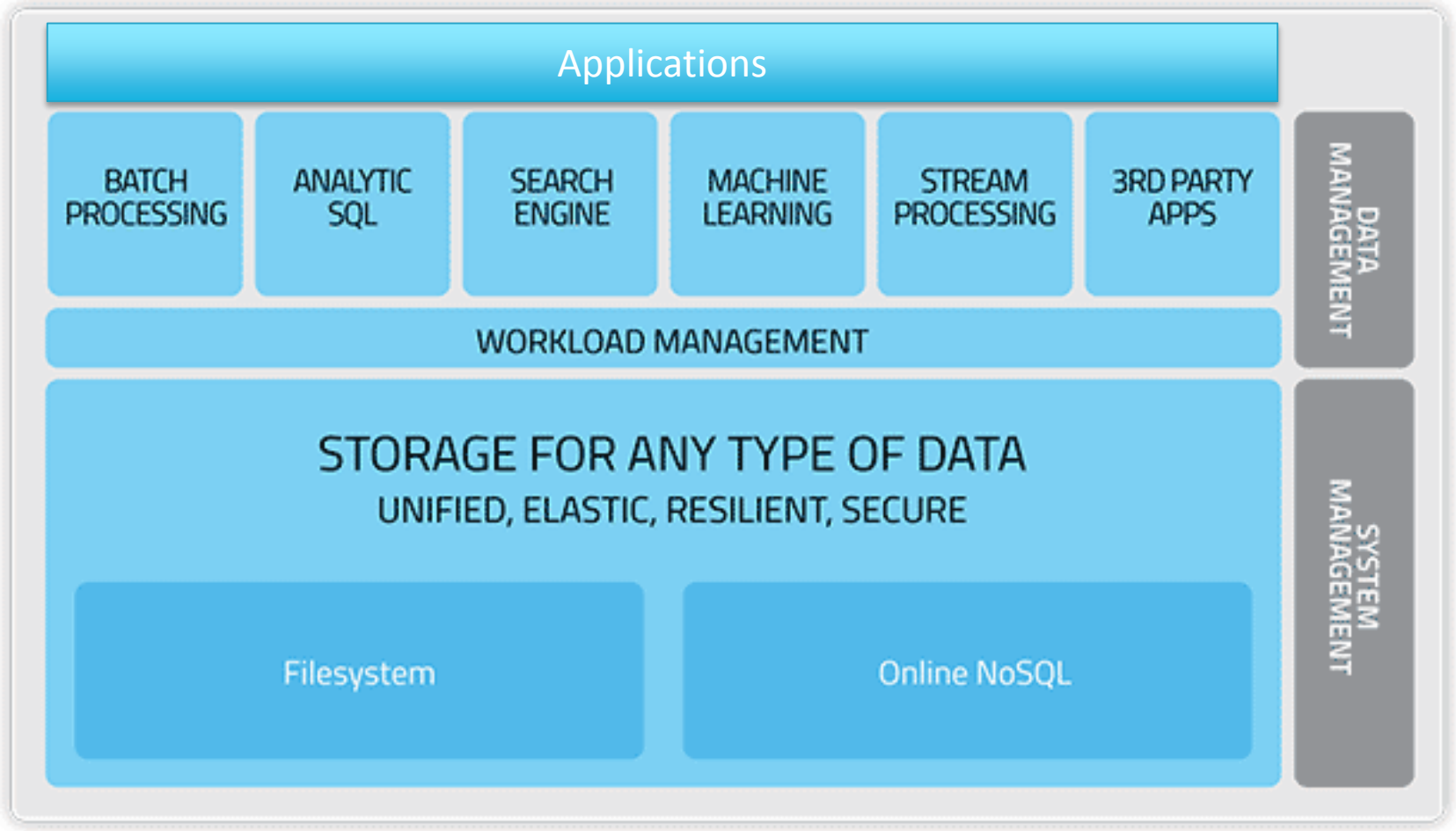
Information & data accessible by all for insight using leading tools and apps

Enterprise Data Hub
Unified Data Management Infrastructure

Ingest All Data
Any Type
Any Scale
From Any Source



Hadoop et al Enabling an EDH



From the Monolith to Micro-Services

Randy Shoup
@randyshoup
linkedin.com/in/randyshoup

The Monolithic Architecture

- Single, vertically-integrated unit
- "The System"

The Monolithic Architecture

Pros

- Simple at first
- In-process latencies
- Single codebase, deploy unit
- Resource-efficient at small scale

Cons

- Coordination overhead as team grows
- Poor enforcement of modularity
- Poor scaling (vertical only)
- All-or-nothing deploy (downtime, failures)
- Long build times

The Monolithic Architecture, v2

- Set of monolithic tiers
- "The front-end", "The app server", "The database"

The Monolithic Database

Pros

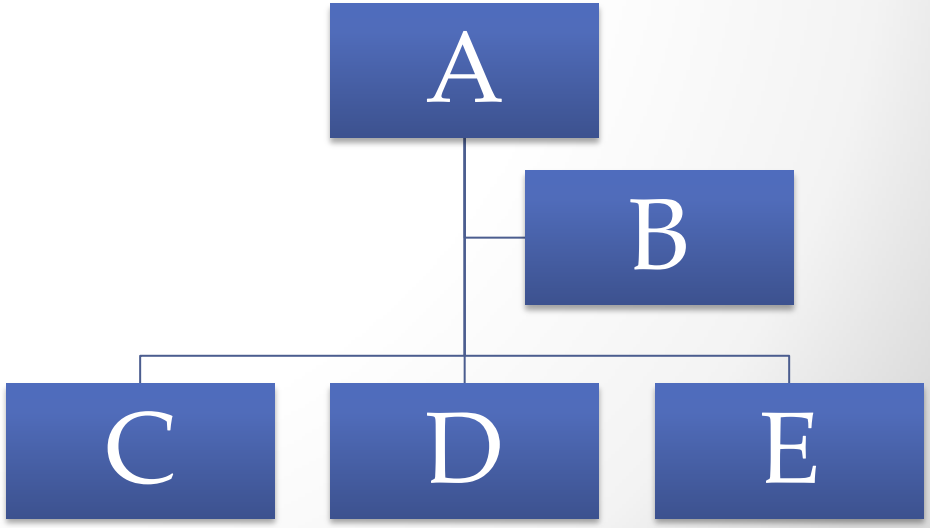
- Simple at first
- Join queries are easy
- Single schema, deployment
- Resource-efficient at small scale

Cons

- Coupling over time
- Poor scaling and redundancy (all-or-nothing, vertical only)
- Difficult to tune properly
- All-or-nothing schema management

Micro-Services

- Single-purpose
- Simple, well-defined interface
- Modular and independent
- More graph of relationships than tiers
- Fullest expression of modularity and encapsulation



Micro-Services

Pros

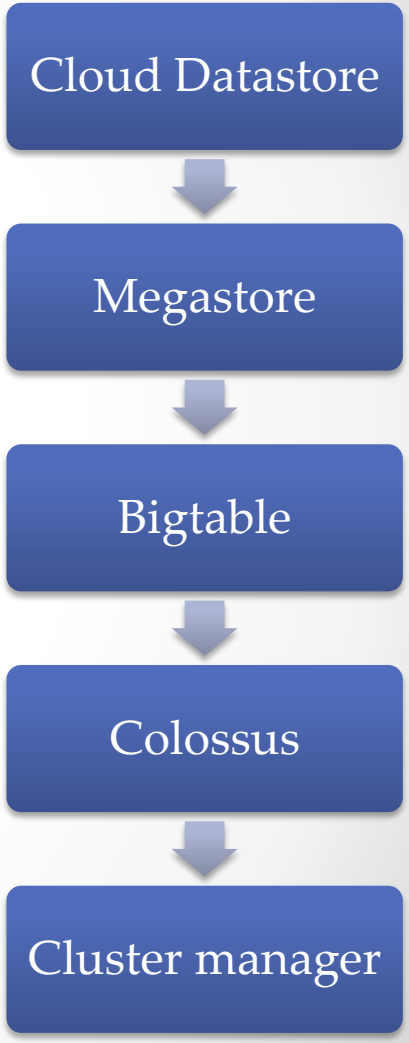
- Each unit is simple
- Independent scaling and performance
- Independent testing and deployment
- Can optimally tune performance (caching, replication, etc.)

Cons

- Many cooperating units
- Many small repos
- Requires more sophisticated tooling and dependency management
- Network latencies

Google Cloud Datastore

- Cloud Datastore: NoSQL service
  - Highly scalable and resilient
  - Strong transactional consistency
  - SQL-like rich query capabilities
- Megastore: geo-scale structured database
  - Multi-row transactions
  - Synchronous cross-datacenter replication
- Bigtable: cluster-level structured storage
  - (row, column, timestamp) -> cell contents
- Colossus: next-generation clustered file system
  - Block distribution and replication
- Cluster management infrastructure
  - Task scheduling, machine assignment



Pro-Tips: Building a Micro-Service

- Common Chassis
  - Make it trivially easy to build and maintain a service
- Define Service Interface (Formally!)
  - Propose, Discuss, Agree
- Prototype Implementation
  - Simplest thing that could possibly work
  - Client can integrate with prototype
  - Implementor can learn what works and what does not
- Real Implementation
  - Throw away the prototype (!)
- ➔ Rinse and Repeat

Transition to Service Relationships

- Vendor – Customer Relationship
  - Friendly and cooperative, but structured
  - Clear ownership and division of responsibility
  - Customer can choose to use service or not (!)
- Service-Level Agreement (SLA)
  - Promise of service levels by the provider
  - Customer needs to be able to rely on the service, like a utility
- Charging and Cost Allocation
  - Charge customers for *usage* of the service
  - Aligns economic incentives of customer and provider
  - Motivates both sides to optimize

Why Enterprises are Embracing the Cloud

Randy Shoup
@randyshoup
linkedin.com/in/randyshoup

Embracing the Cloud (The Obvious)

- Provisioning Speed
  - Minutes, not weeks
  - Autoscaling in response to load
- Near-Infinite Capacity
  - No need to predict and plan for growth
  - No need to defensively overprovision
- Pay For What You Use
  - No "utilization risk" from owning / renting
  - If it's not in use, spin it down

Embracing the Cloud (The Less Obvious)

- Instance Optimization Opportunities
  - Instance shapes to fit most parts of the solution space (compute-intensive, IO-intensive, etc.)
  - If the shape does not fit, try another
- Service Quality
  - Amazon and Google know how to run data centers
  - Battle-tested and highly automated
  - World-class networking, both cluster fabric and external peering
- Unstoppable Economics
  - Almost impossible to beat Google / Amazon buying power or operating efficiencies
  - 2010s in computing are like 1910s in electric power

"Soon it will be just as common to run your own data center as it is to run your own electric power generation"

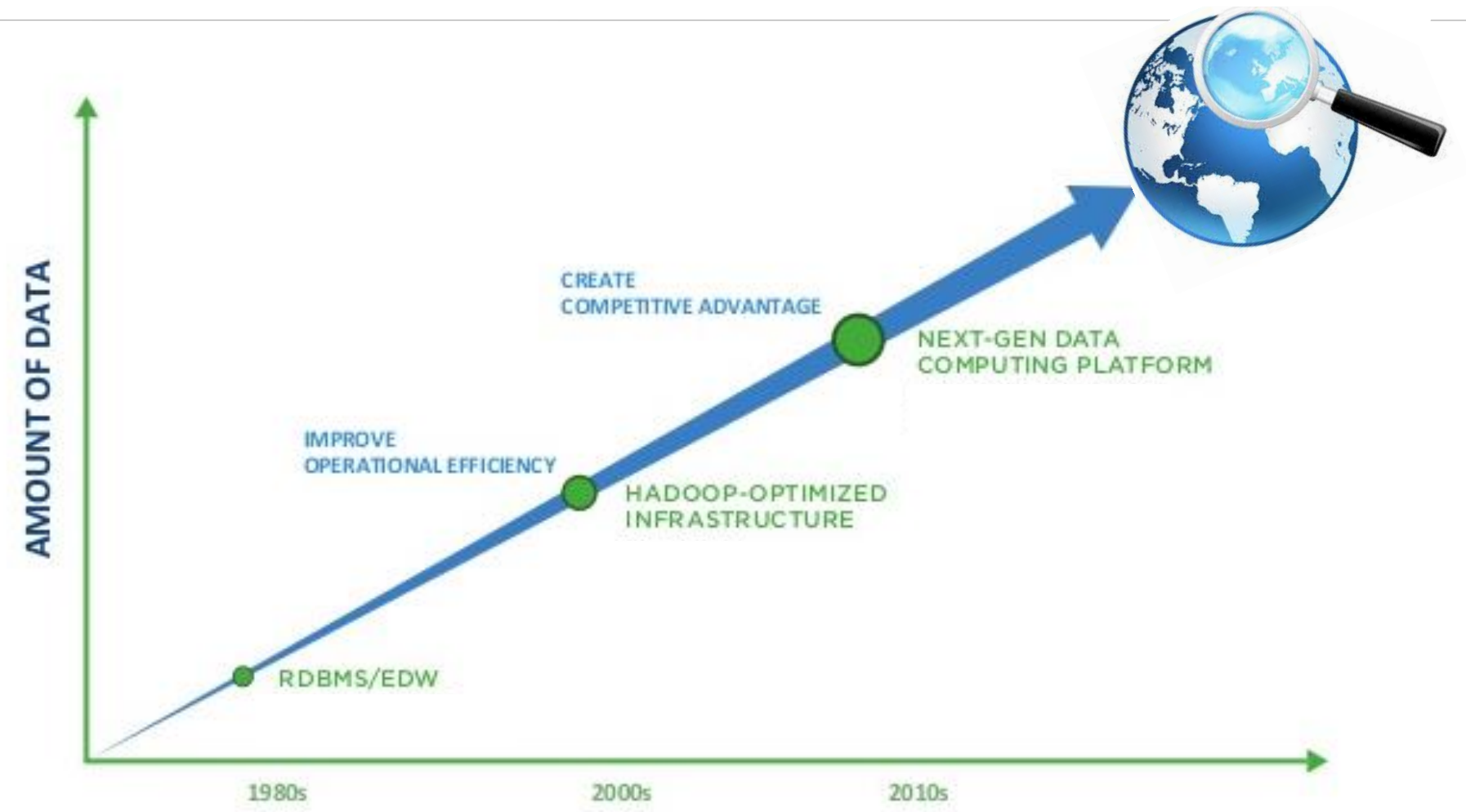
-- me

Big Data Fuels IT Architecture Evolution

@EvaAndreasson, Cloudera

Data Re-Thinking Drivers

Where we are Heading…

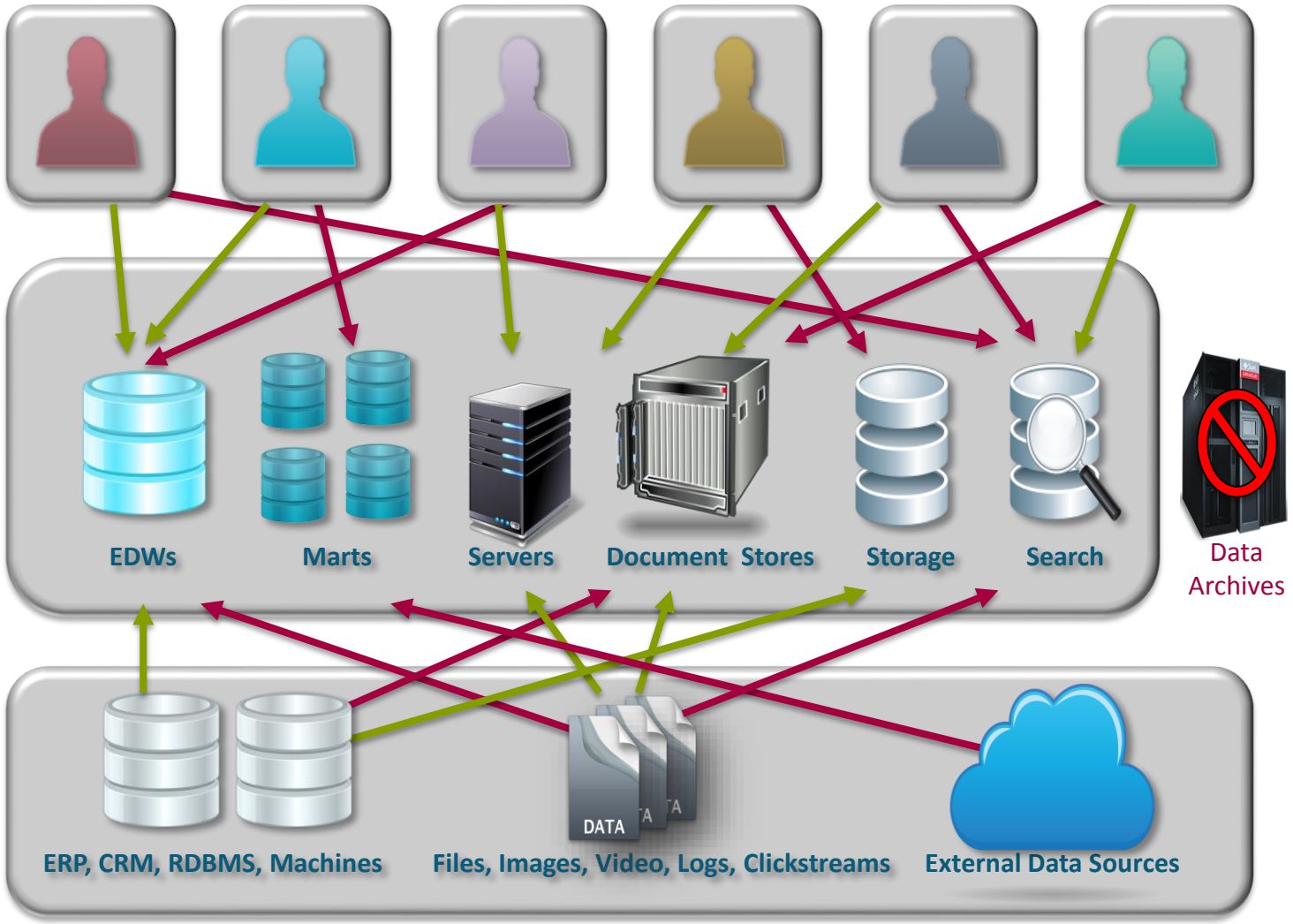


The Need to Rethink Data Architecture

Thousands of Employees & Lots of Inaccessible Information

Heterogeneous Legacy IT Infrastructure

Silos of Multi-Structured Data Difficult to Integrate

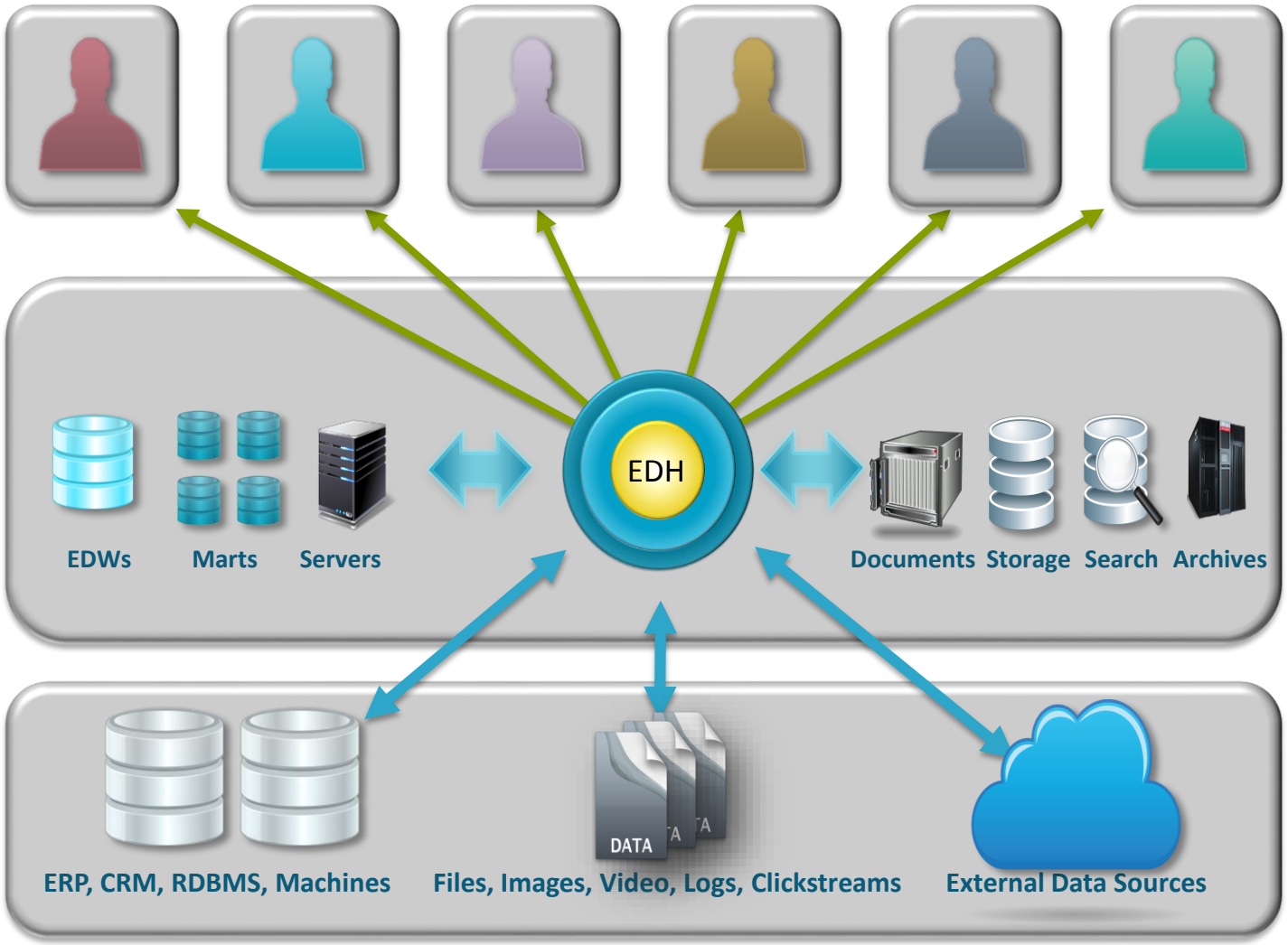


New Category: The Enterprise Data Hub (EDH)

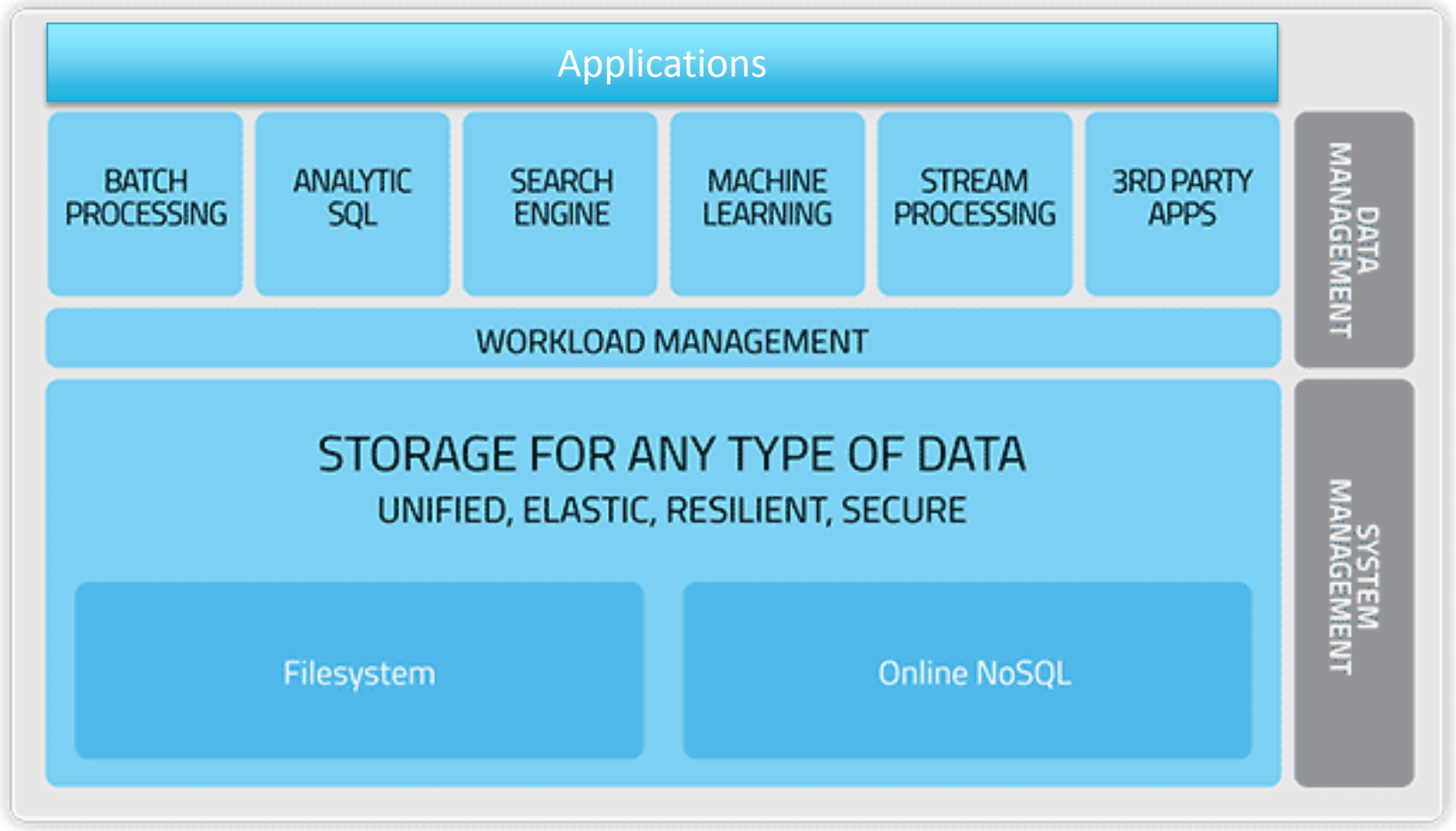
Information & data accessible by all for insight using leading tools and apps

Enterprise Data Hub
Unified Data Management Infrastructure

Ingest All Data
Any Type
Any Scale
From Any Source



Hadoop et al Enabling an EDH



Why Enterprises are Embracing the Cloud

Randy Shoup
@randyshoup
linkedin.com/in/randyshoup

Embracing the Cloud (The Obvious)

- Provisioning Speed
  - Minutes, not weeks
  - Autoscaling in response to load
- Near-Infinite Capacity
  - No need to predict and plan for growth
  - No need to defensively overprovision
- Pay For What You Use
  - No "utilization risk" from owning / renting
  - If it's not in use, spin it down

Embracing the Cloud (The Less Obvious)

- Instance Optimization Opportunities
  - Instance shapes to fit most parts of the solution space (compute-intensive, IO-intensive, etc.)
  - If the shape does not fit, try another
- Service Quality
  - Amazon and Google know how to run data centers
  - Battle-tested and highly automated
  - World-class networking, both cluster fabric and external peering
- Unstoppable Economics
  - Almost impossible to beat Google / Amazon buying power or operating efficiencies
  - 2010s in computing are like 1910s in electric power

"Soon it will be just as common to run your own data center as it is to run your own electric power generation"

-- me

From the Monolith to Micro-Services

Randy Shoup
@randyshoup
linkedin.com/in/randyshoup

The Monolithic Architecture

- Single, vertically-integrated unit
- "The System"

The Monolithic Architecture

Pros

- Simple at first
- In-process latencies
- Single codebase, deploy unit
- Resource-efficient at small scale

Cons

- Coordination overhead as team grows
- Poor enforcement of modularity
- Poor scaling (vertical only)
- All-or-nothing deploy (downtime, failures)
- Long build times

The Monolithic Architecture, v2

- Set of monolithic tiers
- "The front-end", "The app server", "The database"

The Monolithic Database

Pros

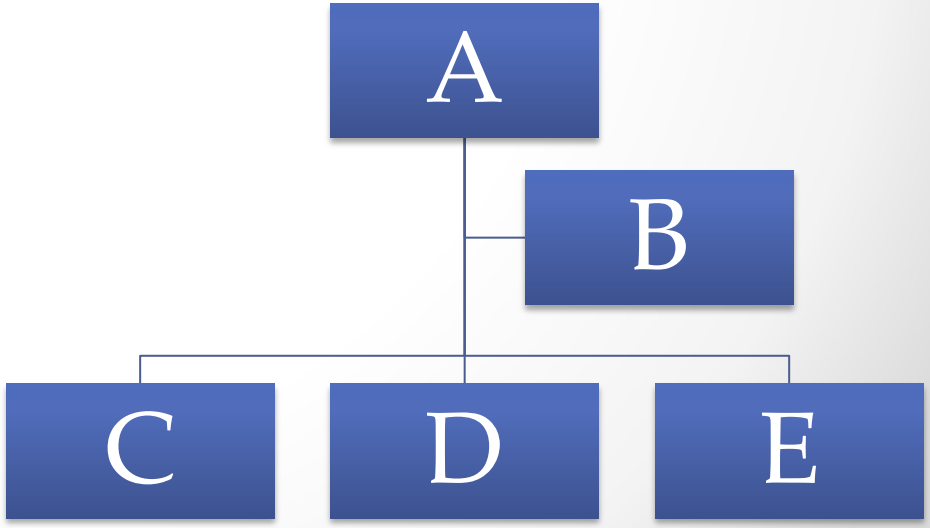
- Simple at first
- Join queries are easy
- Single schema, deployment
- Resource-efficient at small scale

Cons

- Coupling over time
- Poor scaling and redundancy (all-or-nothing, vertical only)
- Difficult to tune properly
- All-or-nothing schema management

Micro-Services

- Single-purpose
- Simple, well-defined interface
- Modular and independent
- More graph of relationships than tiers
- Fullest expression of modularity and encapsulation



Micro-Services

Pros

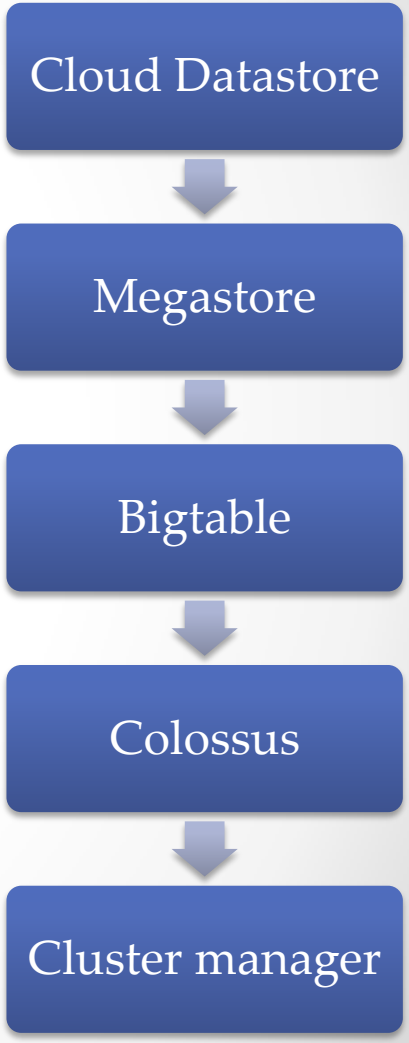
- Each unit is simple
- Independent scaling and performance
- Independent testing and deployment
- Can optimally tune performance (caching, replication, etc.)

Cons

- Many cooperating units
- Many small repos
- Requires more sophisticated tooling and dependency management
- Network latencies

Google Cloud Datastore

- Cloud Datastore: NoSQL service
  - Highly scalable and resilient
  - Strong transactional consistency
  - SQL-like rich query capabilities
- Megastore: geo-scale structured database
  - Multi-row transactions
  - Synchronous cross-datacenter replication
- Bigtable: cluster-level structured storage
  - (row, column, timestamp) -> cell contents
- Colossus: next-generation clustered file system
  - Block distribution and replication
- Cluster management infrastructure
  - Task scheduling, machine assignment



Pro-Tips: Building a Micro-Service

- Common Chassis
  - Make it trivially easy to build and maintain a service
- Define Service Interface (Formally!)
  - Propose, Discuss, Agree
- Prototype Implementation
  - Simplest thing that could possibly work
  - Client can integrate with prototype
  - Implementor can learn what works and what does not
- Real Implementation
  - Throw away the prototype (!)
- ➔ Rinse and Repeat

Transition to Service Relationships

- Vendor – Customer Relationship
  - Friendly and cooperative, but structured
  - Clear ownership and division of responsibility
  - Customer can choose to use service or not (!)
- Service-Level Agreement (SLA)
  - Promise of service levels by the provider
  - Customer needs to be able to rely on the service, like a utility
- Charging and Cost Allocation
  - Charge customers for *usage* of the service
  - Aligns economic incentives of customer and provider
  - Motivates both sides to optimize