CloudClustering

Toward an Iterative Data Processing Pattern on the Cloud

Ankur Dave*, Wei Lu†, Jared Jackson†, Roger Barga†

*UC Berkeley
†Microsoft Research

Ankur Dave

Wei Lu

Jared Jackson

Roger Barga

Microsoft® Research

Background

MapReduce and its successors reimplement communication and storage

But cloud services like EC2 and Azure natively provide these utilities

Can we build an efficient distributed application using only the primitives of the cloud?

Design Goals

- **Efficient**
- **Fault tolerant**
- **Cloud-native**

Outline

Windows Azure

K-Means clustering algorithm

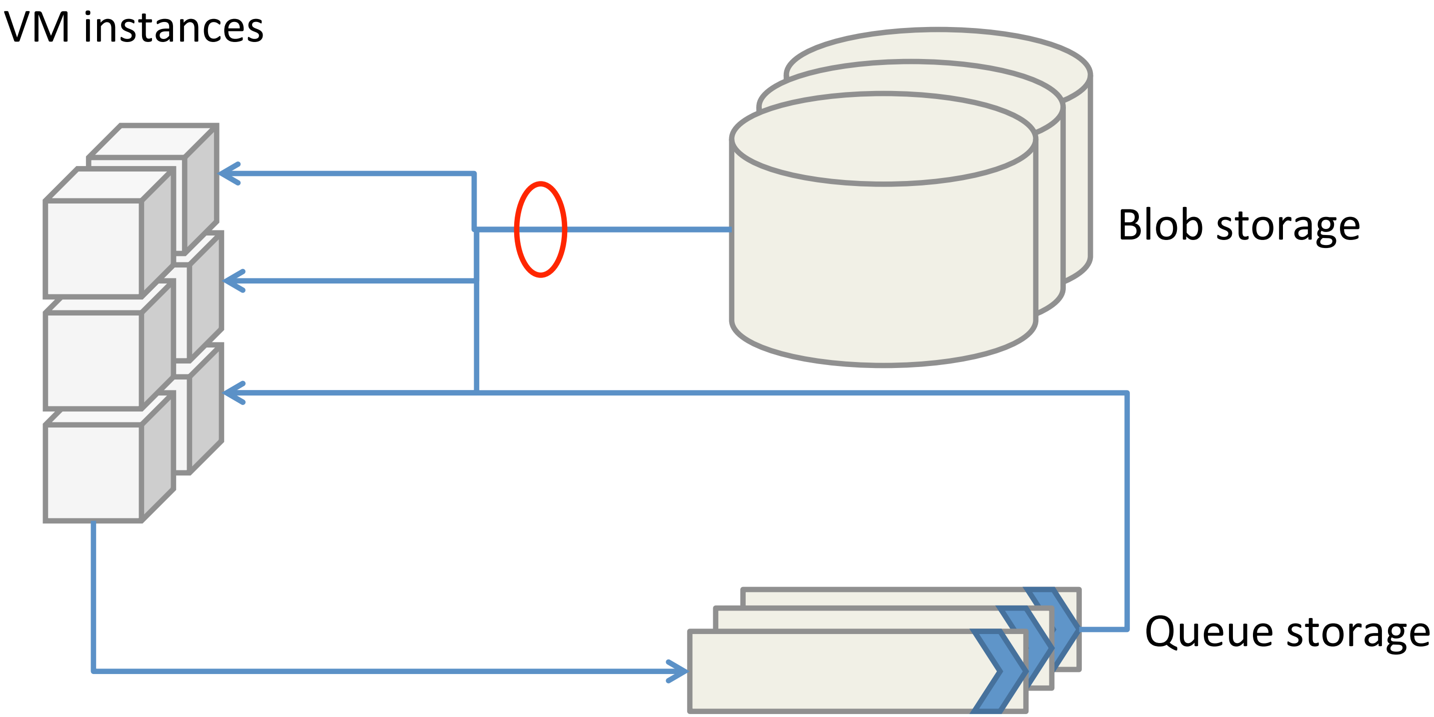
CloudClustering architecture

- Data locality
- Buddy system

Evaluation

Related work

Windows Azure



Outline

Windows Azure

K-Means clustering algorithm

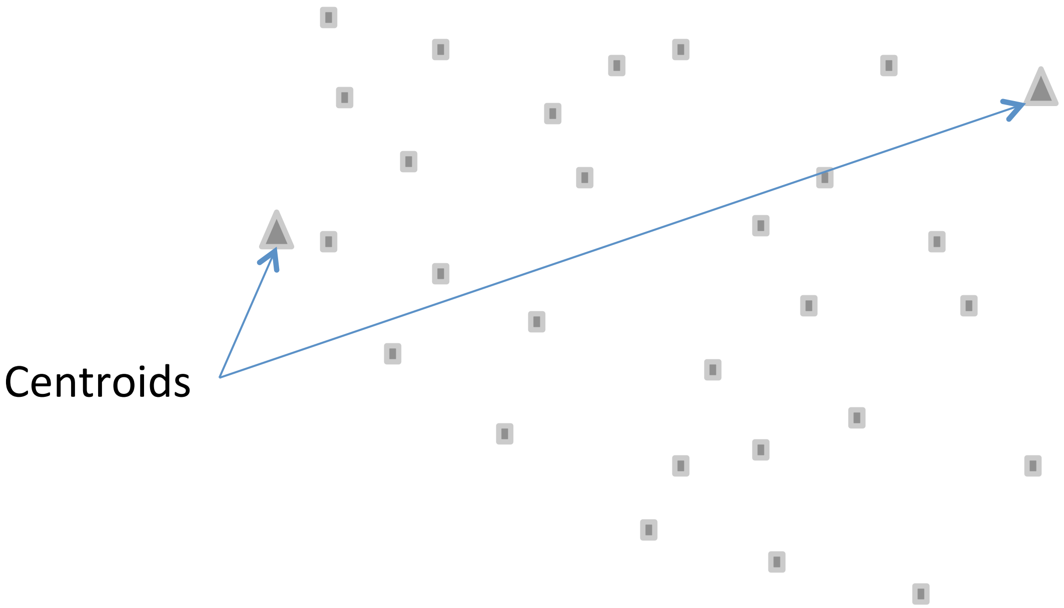
CloudClustering architecture

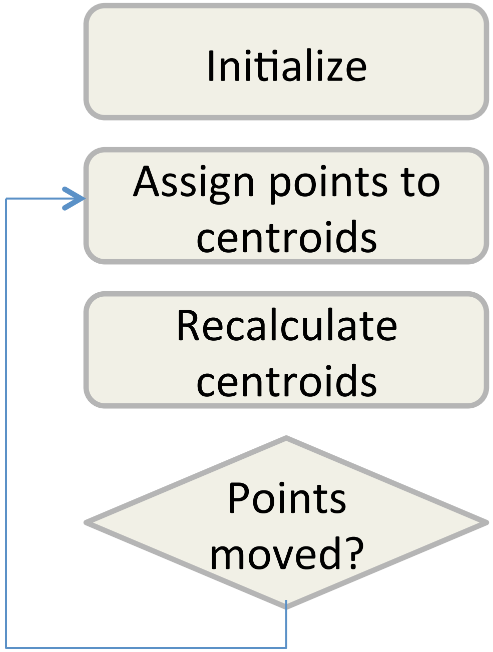
- Data locality
- Buddy system

Evaluation

Related work

K-Means algorithm




Initialize

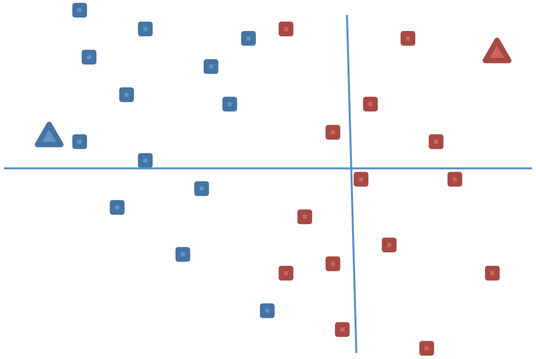
Assign points to centroids

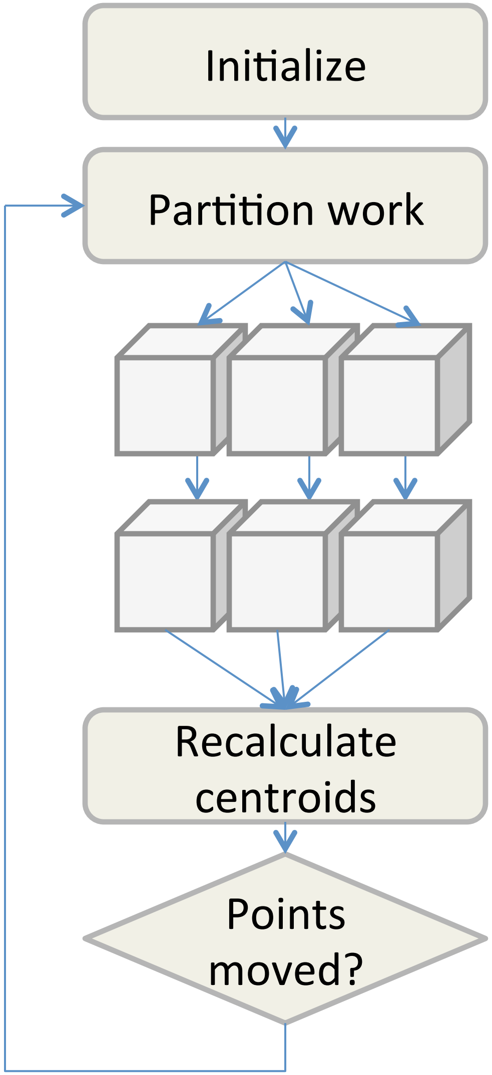
Recalculate centroids

Points moved?

Iteratively groups n points into k clusters

K-Means algorithm





Initialize

Partition work

Assign points to centroids

Compute partial sums

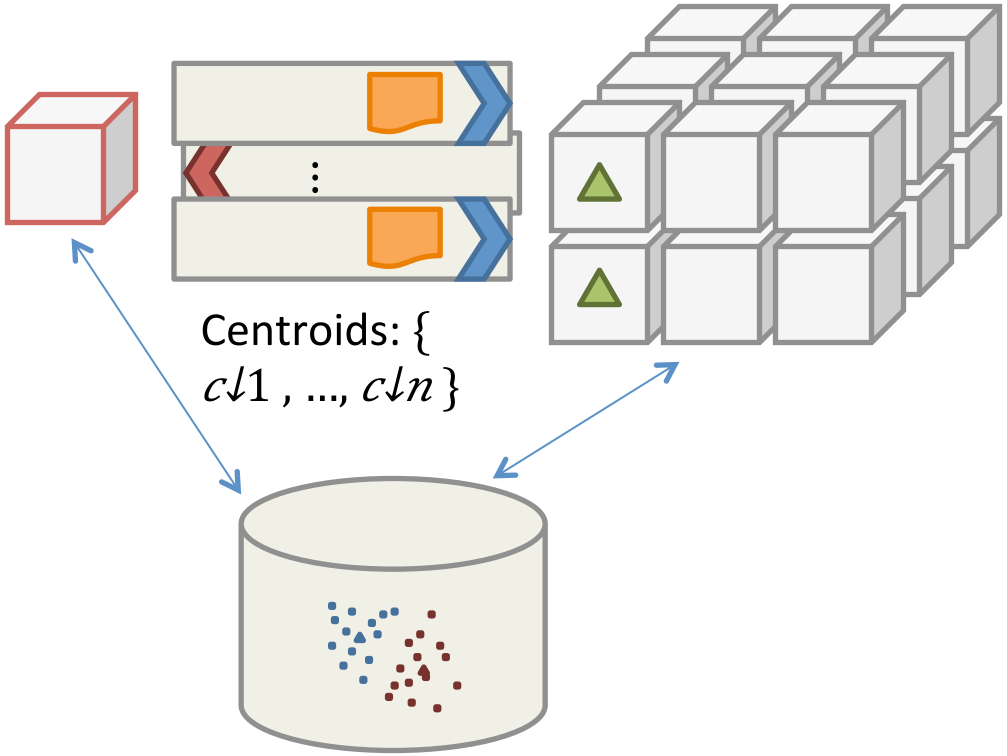
Recalculate centroids

Points moved?

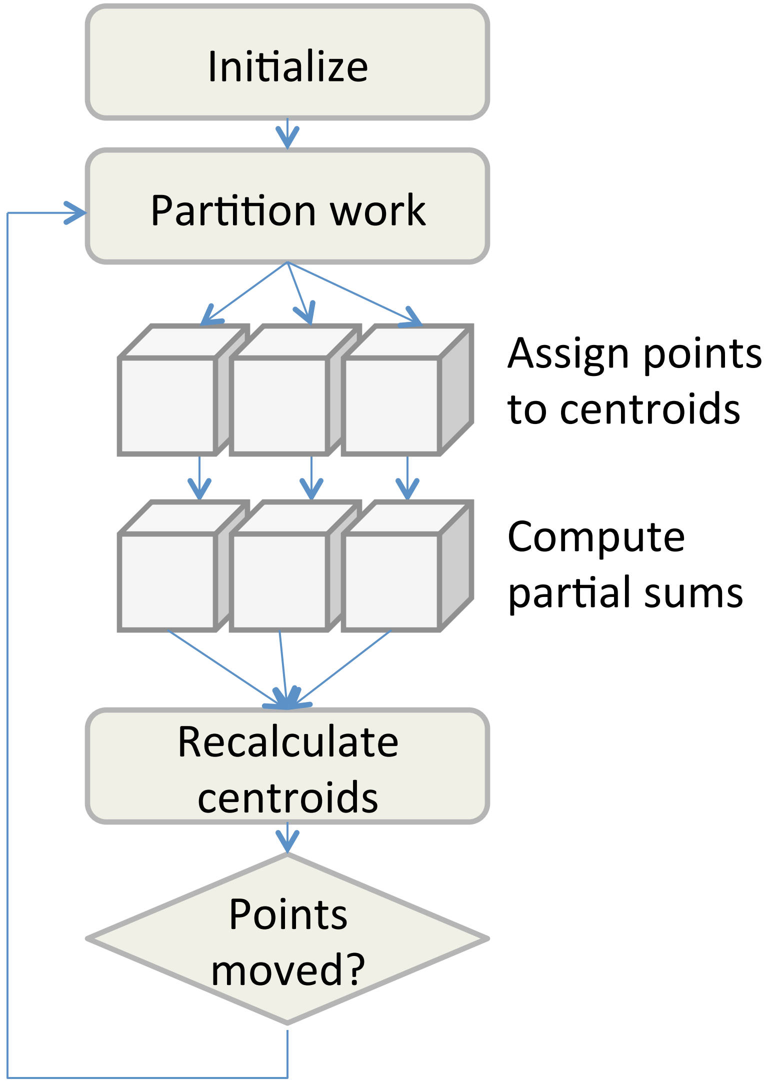
Outline

- Windows Azure
- K-Means clustering algorithm
- **CloudClustering architecture**
  - Data locality
  - Buddy system
- Evaluation
- Related work

Architecture



Centroids: { c↓1 , ..., c↓n }



Initialize

Partition work

Assign points to centroids

Compute partial sums

Recalculate centroids

Points moved?

Outline

Windows Azure

K-Means clustering algorithm

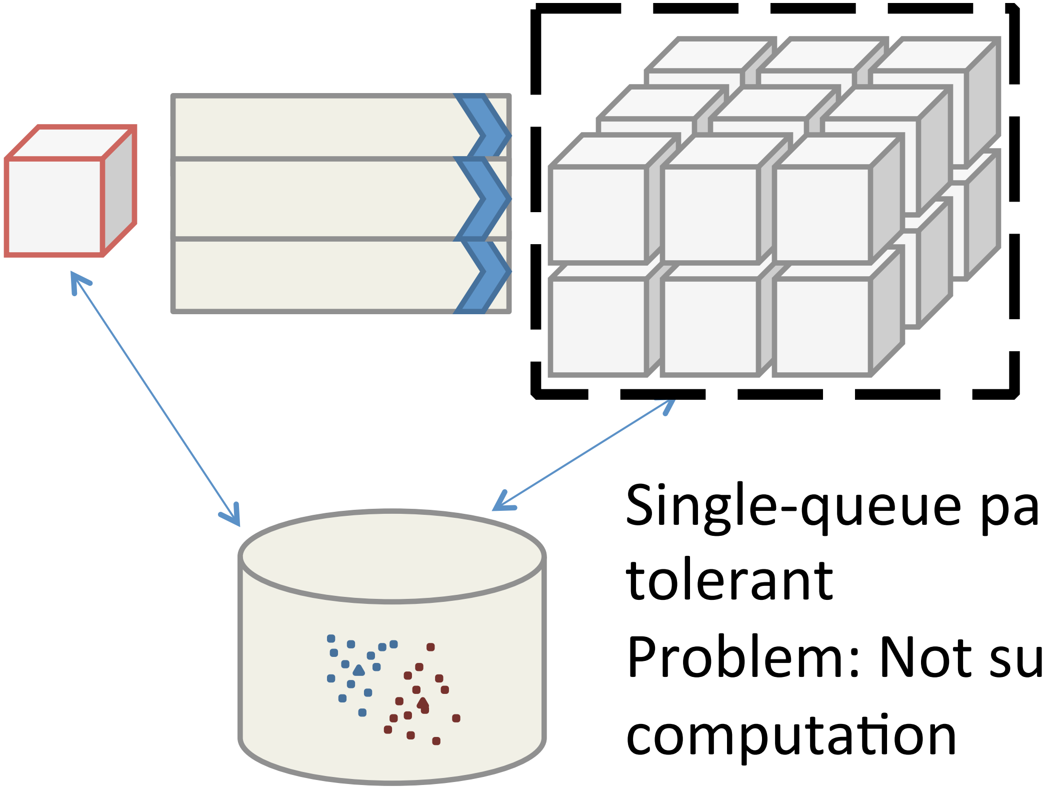
CloudClustering architecture

- **Data locality**
- Buddy system

Evaluation

Related work

Data locality



Single-queue pattern is naturally fault-tolerant

Problem: Not suitable for iterative computation

Multiple-queue pattern unlocks data locality

Tradeoff: Complicates fault tolerance

Outline

Windows Azure

K-Means clustering algorithm

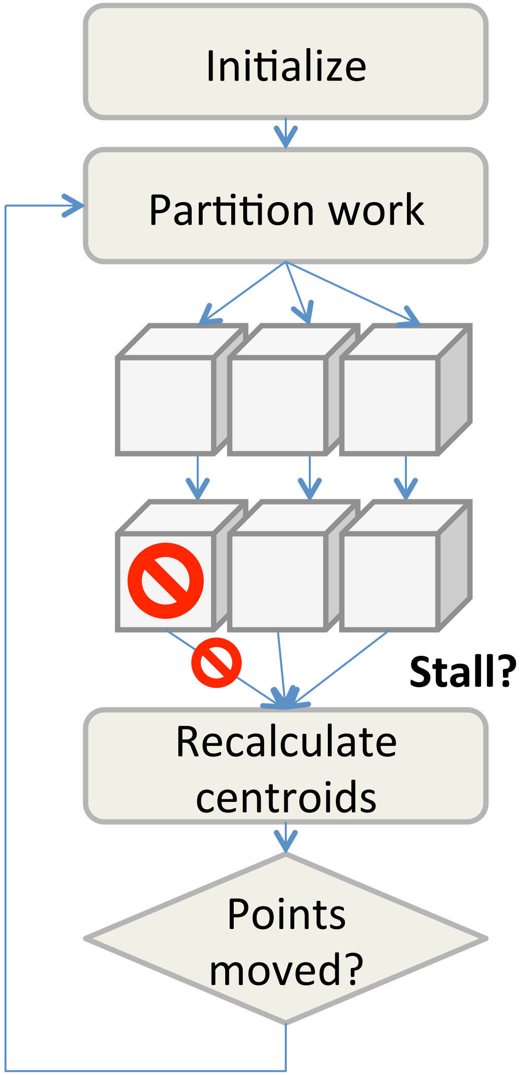
CloudClustering architecture

- Data locality
- **Buddy system**

Evaluation

Related work

Handling failure



Initialize

Partition work

Assign points to centroids

Compute partial sums

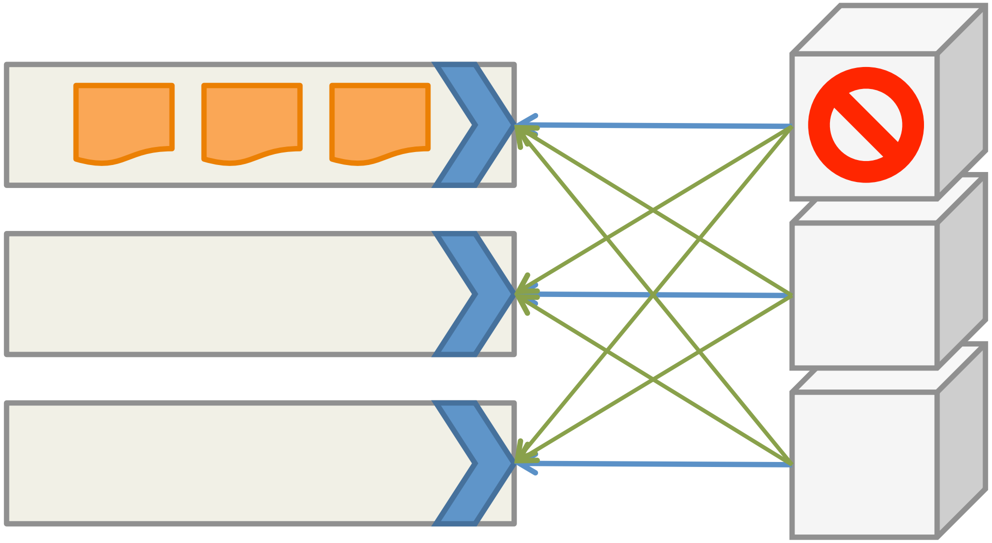
Stall?

Recalculate centroids

Points moved?

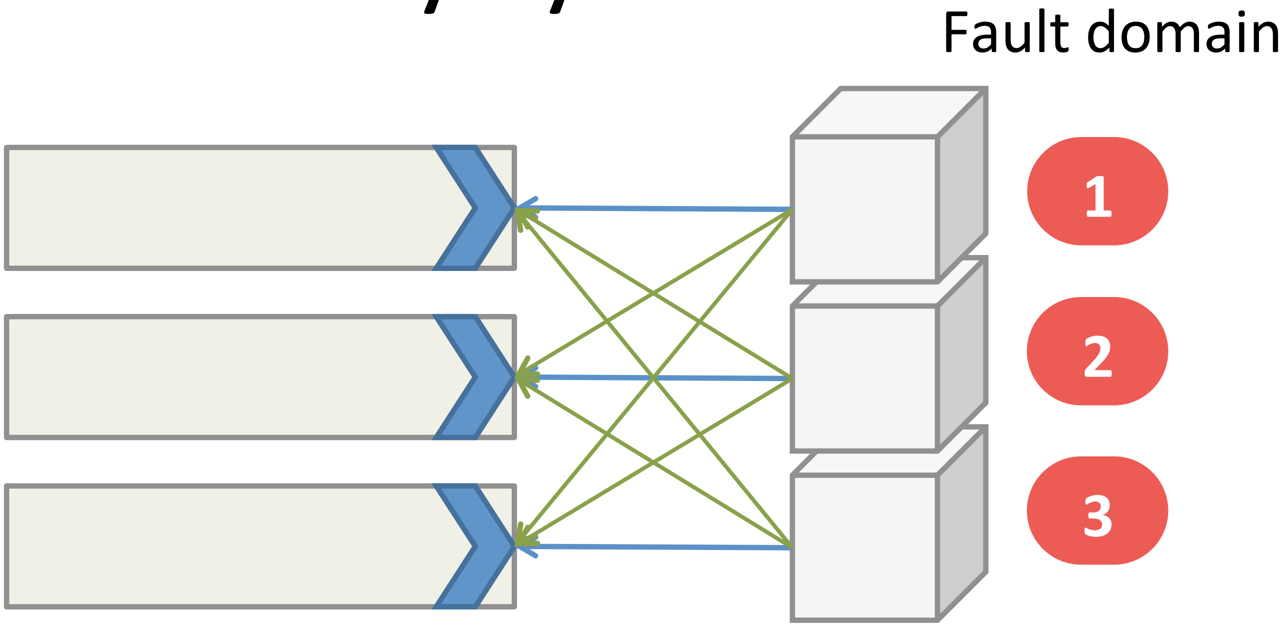
→ **Buddy system**

Buddy system



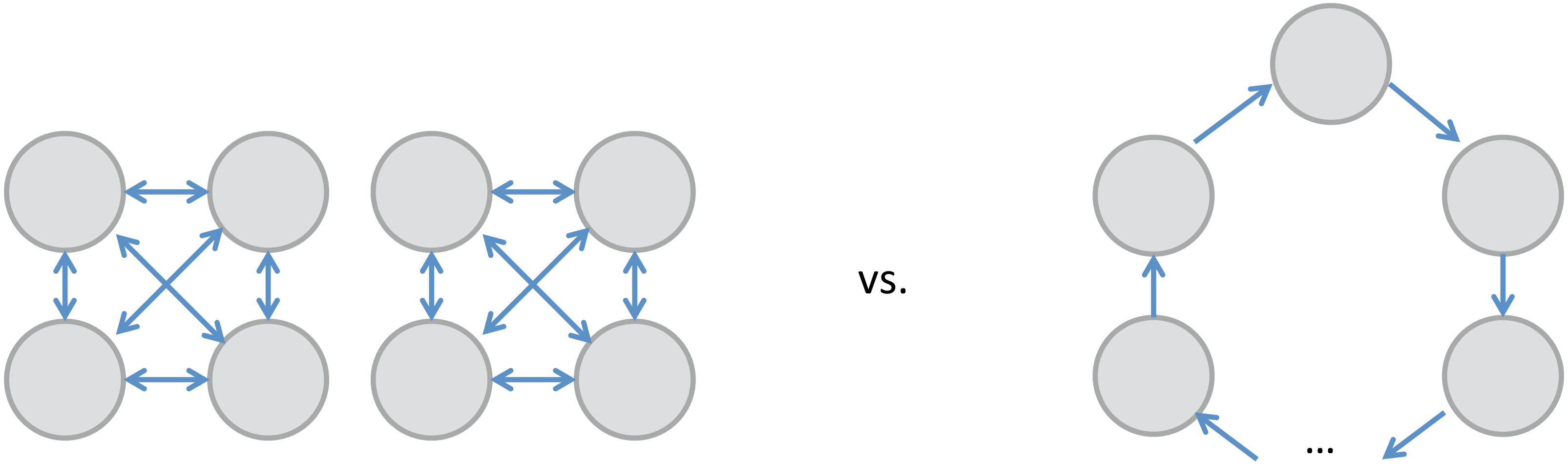
Buddy system provides distributed fault detection and recovery

Buddy system



Spreading buddies across Azure fault domains provides increased resilience to simultaneous failure

Buddy system



Cascaded failure detection reduces communication and improves resilience

Outline

Windows Azure

K-Means clustering algorithm

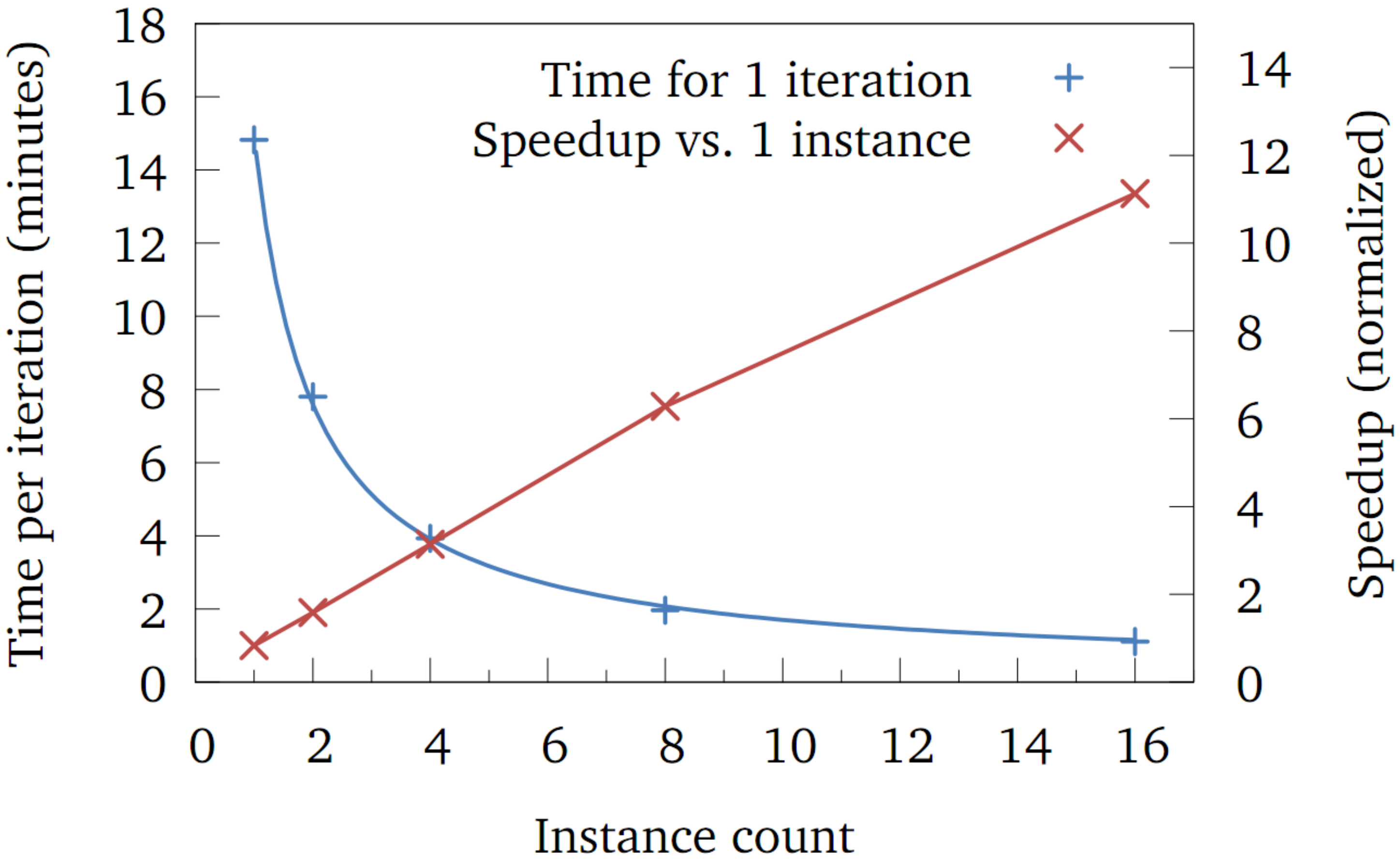
CloudClustering architecture

- Data locality
- Buddy system

Evaluation

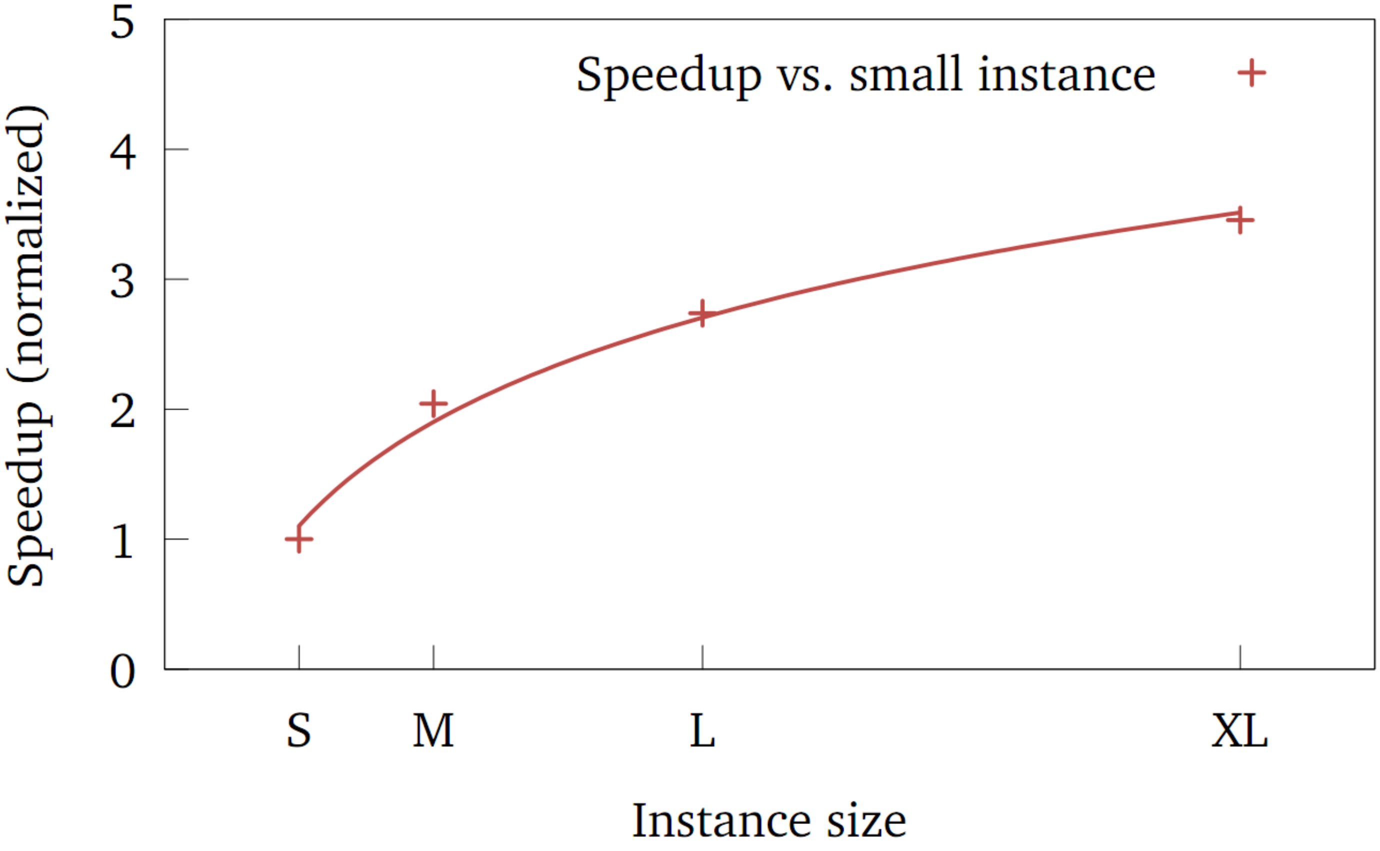
Related work

Evaluation



Linear speedup with instance count

Evaluation



Sublinear speedup with instance size
Reason: I/O bandwidth doesn’t scale

Outline

Windows Azure

K-Means clustering algorithm

CloudClustering architecture

- Data locality
- Buddy system

Evaluation

Related work

Related work

- Existing frameworks (**MapReduce, Dryad, Spark, MPI, …)** all support K-Means, but reimplement reliable communication
- **AzureBlast** built directly on cloud services, but algorithm is not iterative

Conclusions

- CloudClustering shows that it's possible to build *efficient, resilient applications* using only the common cloud services
- Multiple-queue pattern unlocks data locality
- Buddy system provides fault tolerance