Location, Location, Location: Support for Geo-Centric Applications

PIs: Abhishek Chandra and Jon Weissman

Department of Computer Science and Engineering

University of Minnesota

NSF Award: CNS-1619254

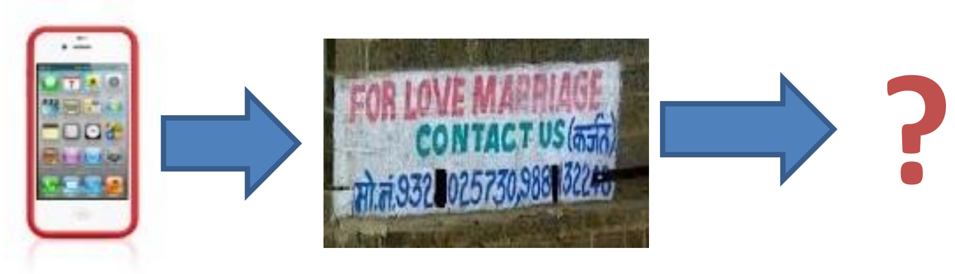
(Start: Oct 2016)

Motivation

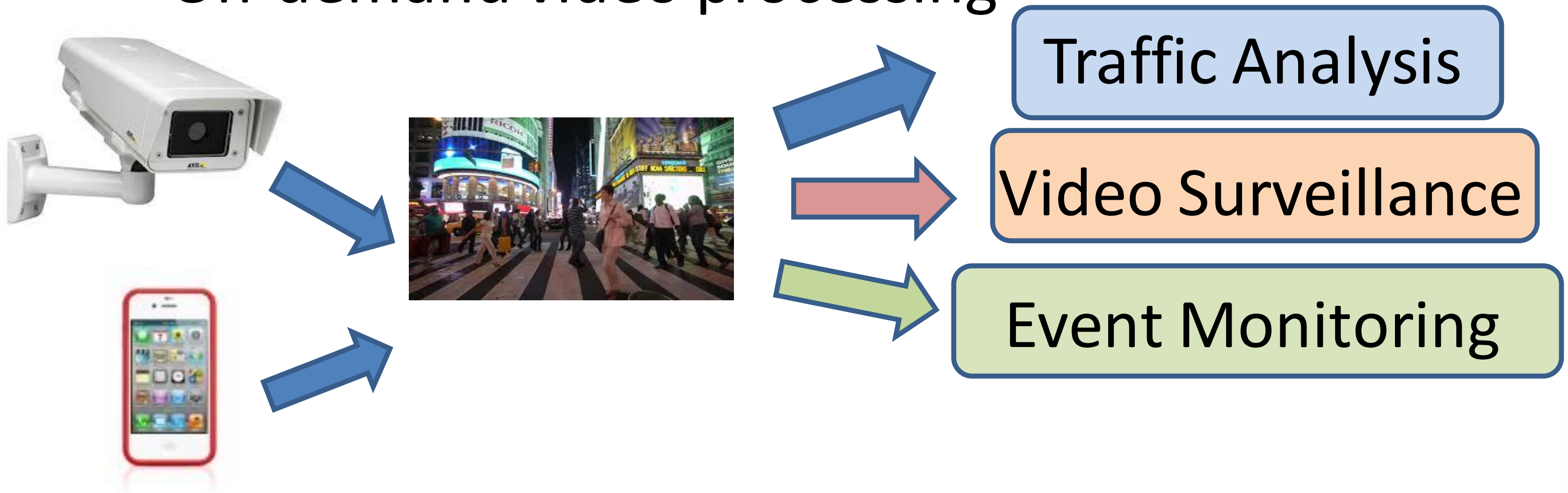
- Geo-distributed mobile devices, sensors, wearable devices

Geo-centric Applications

- Location-dependent: User/data-driven
- Examples:
  - JIT image analysis



  - On-demand video processing



Traffic Analysis

Video Surveillance

Event Monitoring

Geo-centric Applications: Characteristics

- Have diverse resource needs:
  - Compute, storage, data, sensing
- Desire locality to:
  - User: For low latency
  - Data: For efficient processing
- Limitations of traditional approaches:
  - Centralized cloud: High latency, b/w constraint
  - On-device: Limited compute, storage, battery life

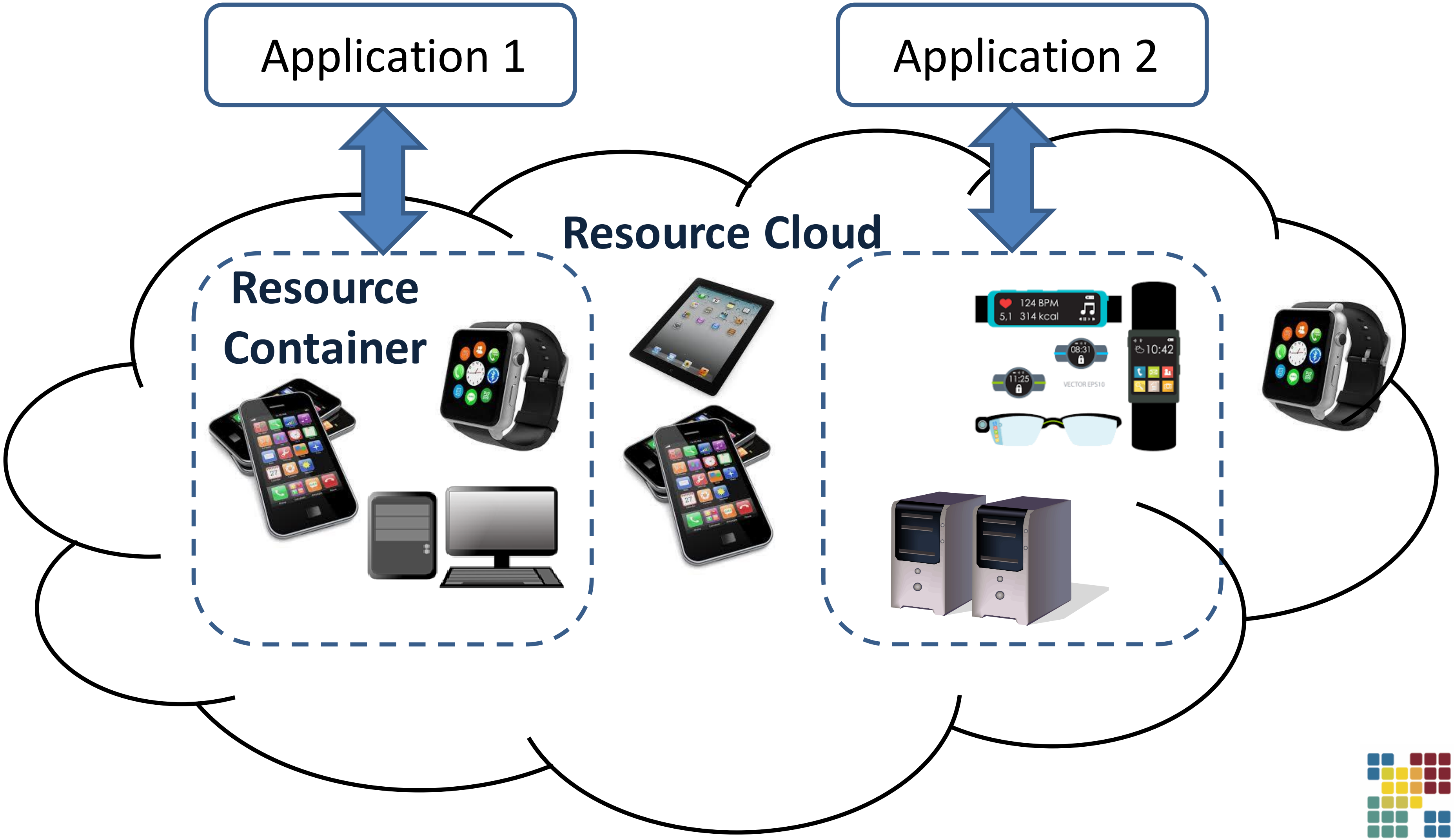
Solution: Use Edge Resources

- Pre-deployed or user-provided
  - Provide compute, storage, sensing capabilities
- Benefits:
  - Good connectivity
  - Large number of users, sensors
  - Powerful, underutilized resources

Challenges

- Mobility: Users or sensors may move
- Unpredictability: Availability and demand of data/resources may vary over time
- Failures: Resources or network may fail
- **Questions:**
  - How to find desired resources on-demand?
  - How can diverse applications easily use them?

Our Approach



Application 1

Application 2

Resource Cloud

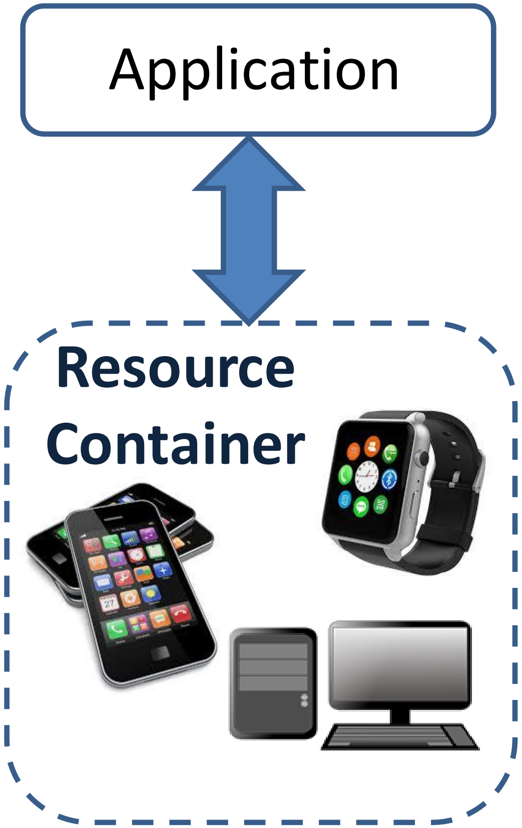
Resource Container

Resource Cloud

- Dynamic collection of available resources
- Location-aware Pub-Sub model:
  - Resource providers publish resources
  - Applications consume resources

Resource Container

- Encapsulates resources for an application
- Simple, general API
  - E.g.: put/get for storage, collect for sensors
- Policy-driven runtime sytem
  - Event-based execution



Application

Resource Container

Talk Outline

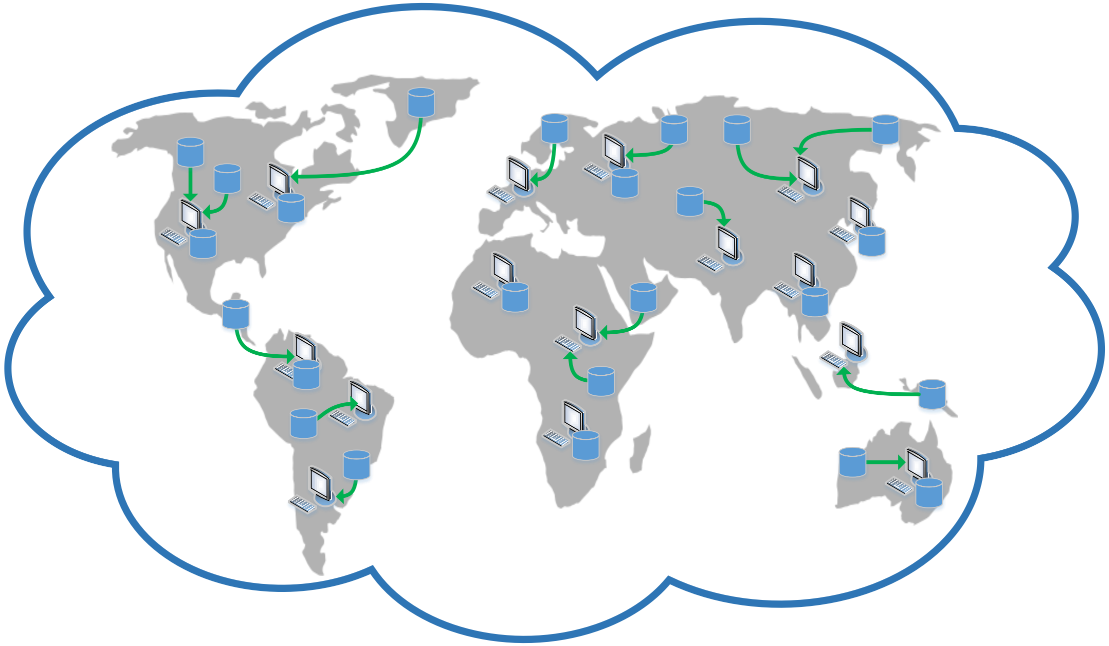
✓Motivation

✓Approach

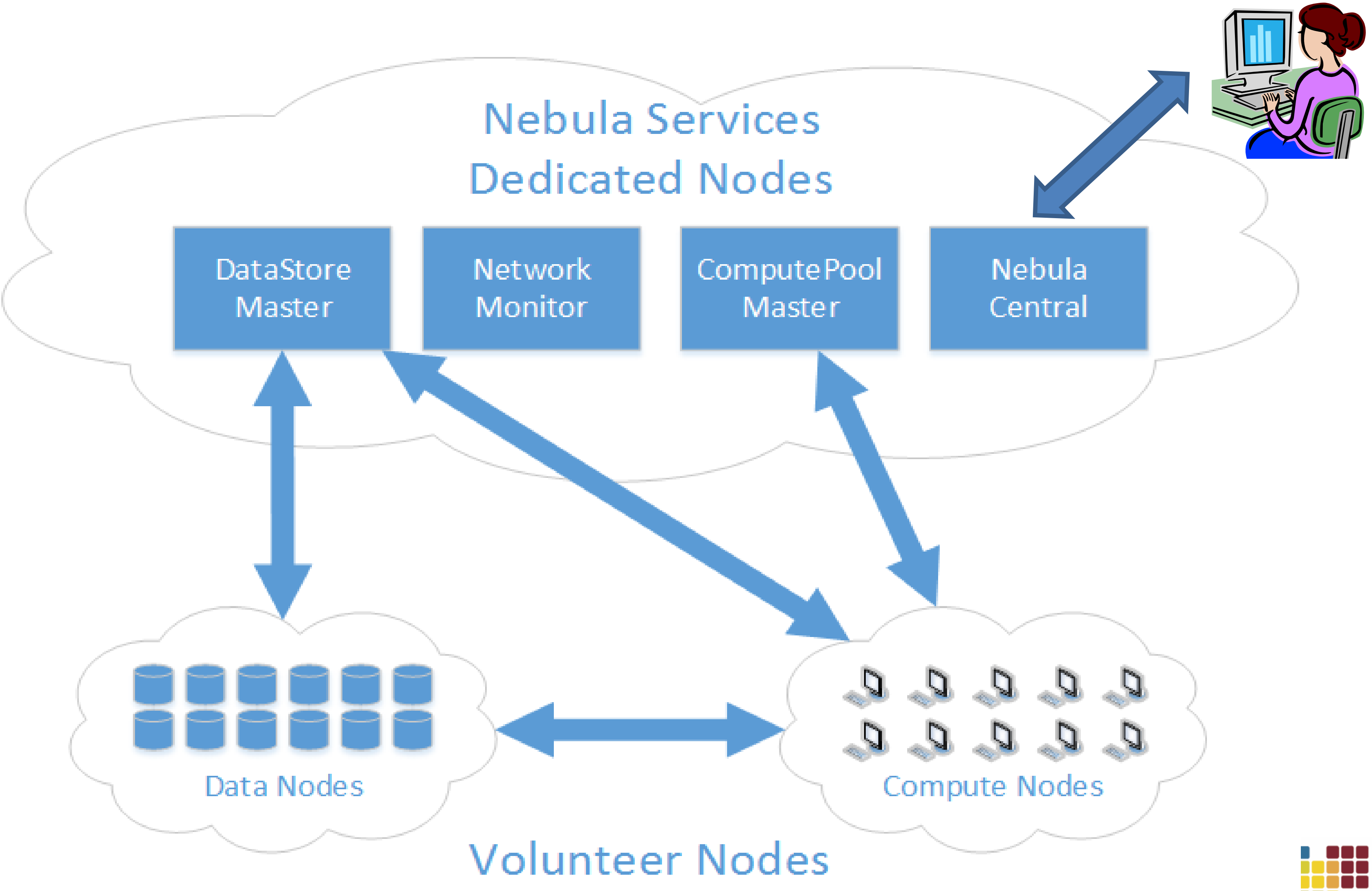
- **Prior Work**
- Ongoing Work
- Conclusion

Prior Work:
Nebula Distributed Edge Cloud

- Exploits volunteer edge computing and storage
- Supports distributed data-intensive applications



Nebula Architecture



Nebula Services

Dedicated Nodes

DataStore Master

Network Monitor

ComputePool Master

Nebula Central

Data Nodes

Compute Nodes

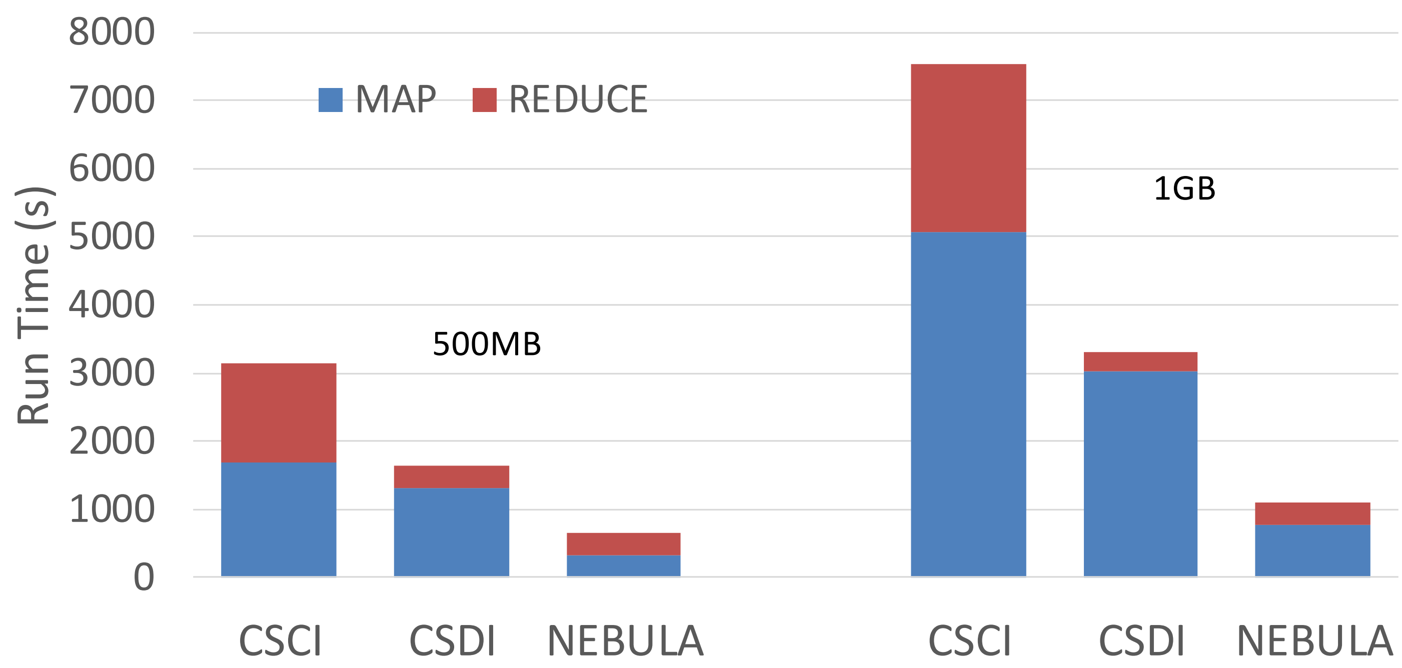
Volunteer Nodes

Locality Awareness

- **Challenge:** Network may be bottleneck
- Locality-aware storage:
  - Data nodes ordered by their locality (b/w, latency, etc.) w.r.t. client
- Locality-aware scheduling
  - Schedule task on a node based on *both* data transfer and computation time

Benefit of Locality-awareness

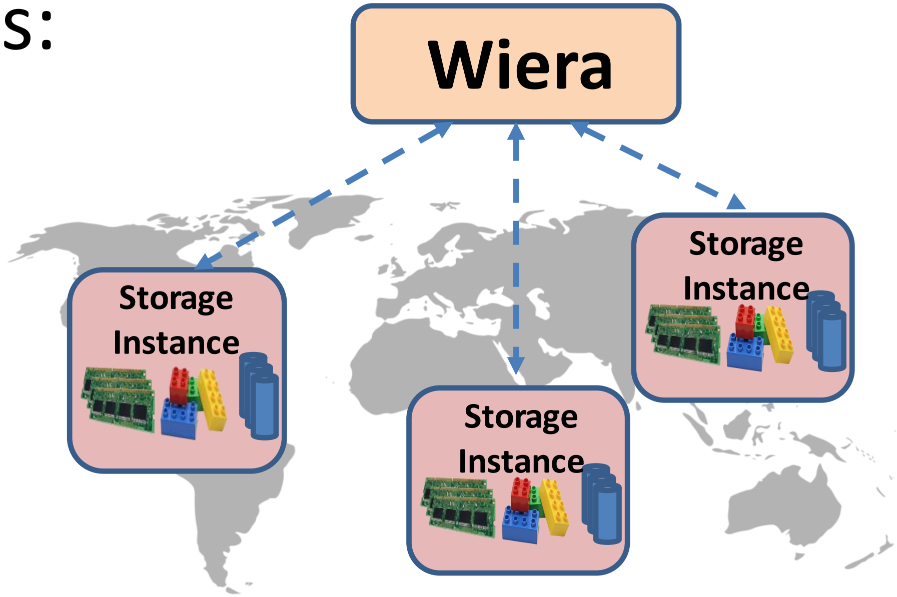
Seteup: PlanetLab; MapReduce Wordcount/Inverted Index



Nebula significantly improves performance via locality-awareness

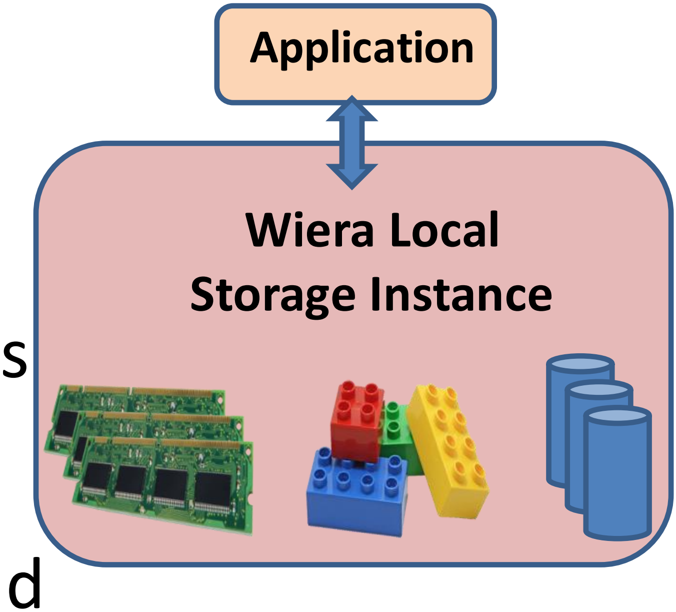
Prior Work:
Wiera Geo-distributed Storage System

- Middleware that supports:
  - Multi-tiered, multi- cloud storage instances
  - Rich array of data management policies
  - Adaptive to network and workload dynamics



Wiera Storage Instance

- Consists of:
  - DC Locations, Storage tiers
  - Storage/Data management policies
- Policies defined using:
  - Events: Action, timer, and threshold
  - Responses: Can be application or storage layer specific
    - E.g.: store, storeOnce, compress, encrypt



Example Wiera Policy

- Desired: Low Latency with periodic writeback

```
Wiera LowLatencyInstance(time t) {
  %two tiers specified with initial sizes
  tier1: { name: Memory, size: 5G };
  tier2: { name: EBS, size: 5G };

  % action event defined to always store data
  % into Memory
  event(insert.into) : response {
      insert.object.dirty = true;
      store(what:insert.object, to:tier1);
  }

  % write back policy: copying data to
  % persistent store on a timer event
  event(time=t) : response {
      copy(what: object.location == tier1 &&
                 object.dirty == true,
           to: tier2);
  }
}
```

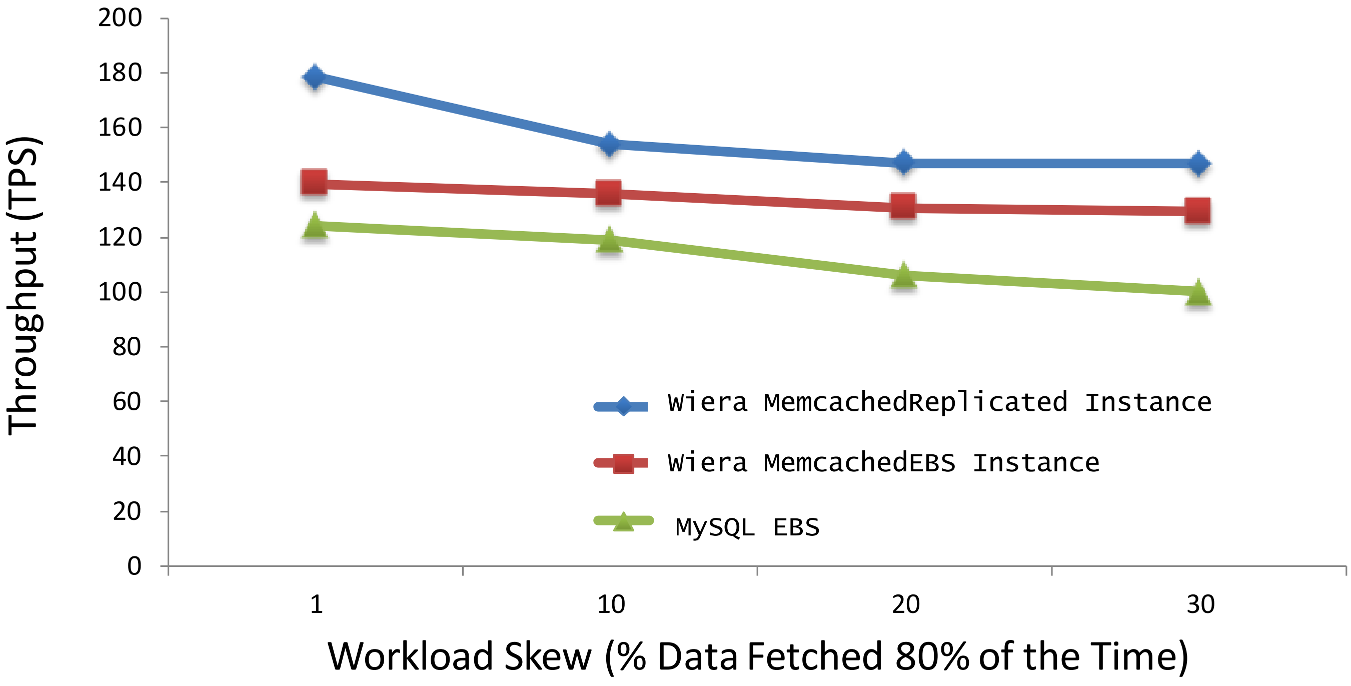
Initial Instance Configuration

ACTION EVENT

TIMER EVENT

Performance Optimization

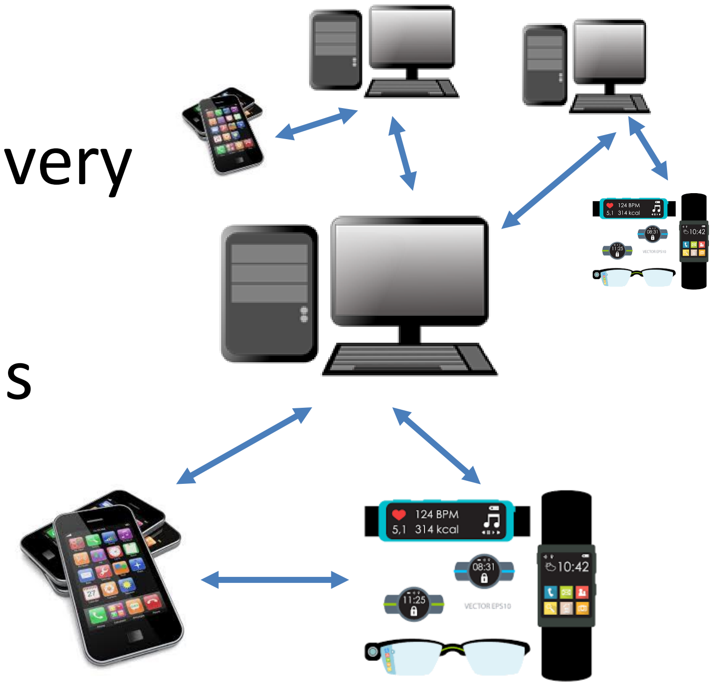
Setup: Amazon AWS; *Unmodified* MySQL



Wiera enables significantly better performance *without* application modifications

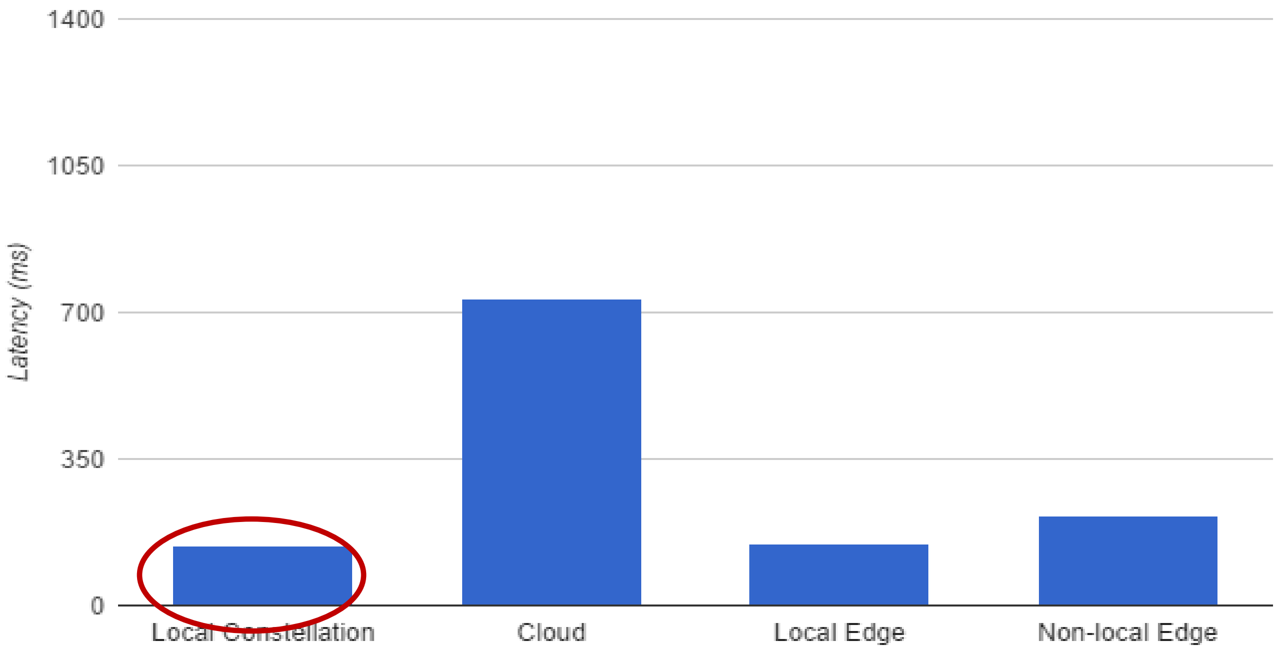
Ongoing Work: Constellation

- Edge-based resource framework for IoT devices
- Supports:
  - Location-aware resource discovery
  - Device-independent API
  - Cross-application optimizations



Preliminary Results

Photon-based temperature sensor, Google Pixel edge device



Constellation achieves low latency with minimal overhead

Ongoing Work

- Support for diverse resource types
  - Compute, storage, sensors, data

- Incentivization
  - Economic models for resource providers

- Richer policy specification and optimization
  - For diverse applications and devices

Concluding Remarks

- Geo-distributed user devices and sensors
- Geo-centric applications:
  - User/data dependent
- Utilizing location-dependent edge resources via:
  - Resource cloud
  - Resource container
- **Acknowledgments:**
  - Students: Albert Jonathan, Zach Leidall, Kwangsung Oh, Ajay Raghavan, Mathew Ryden
  - NSF

Thanks!

http://www.cs.umn.edu/~chandra