MobiSys'11

Odessa: Enabling Interactive Perception Applications on Mobile Devices

Moo-Ryong Ra*, Anmol Sheth+,
Lily Mummert[x], Padmanabhan Pillai',
David Wetherall[o], Ramesh Govindan*

USC ENL, +Technicolor, [x]Google, 'Intel,
[o]University of Washington

Presented by Mohammad Shahrad

Emerging Mobile Perception Applications

GPS

Accelerometer

Sensing

Activity Recognition

Health, Traffic Monitoring

Location-Based Service

Participatory Sensing

Sensing Applications

Emerging Mobile Perception Applications

HD Camera

Sensing

Dual-Core CPU

Computation

Cloud Infrastructure

Communication

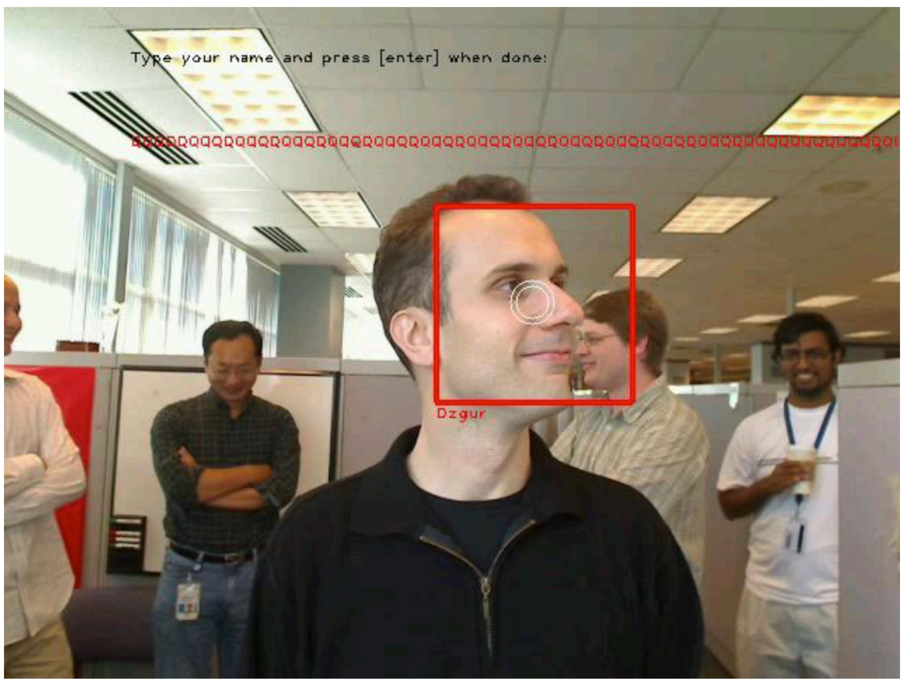
Mobile Interactive Perception Application

Vision-based Interactive Mobile Perception Applications

Face Recognition

Object and Pose Recognition

Gesture Recognition



Common Characteristics

Interactive

- Crisp response time (10 ms ~ 200 ms)

High Data-Rate

- Processing video data of 30 fps

Compute Intensive

- Computer Vision based algorithms

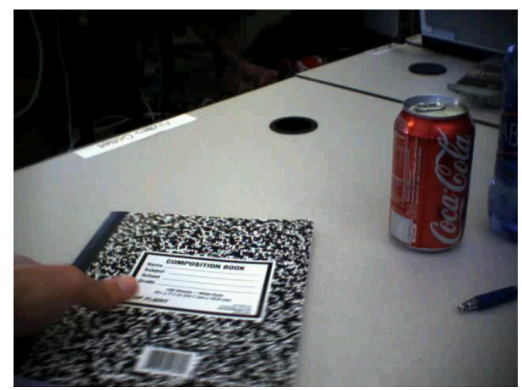
Enabling Mobile Interactive Perception

Performance

Throughput ↑ | **Makespan** ↓

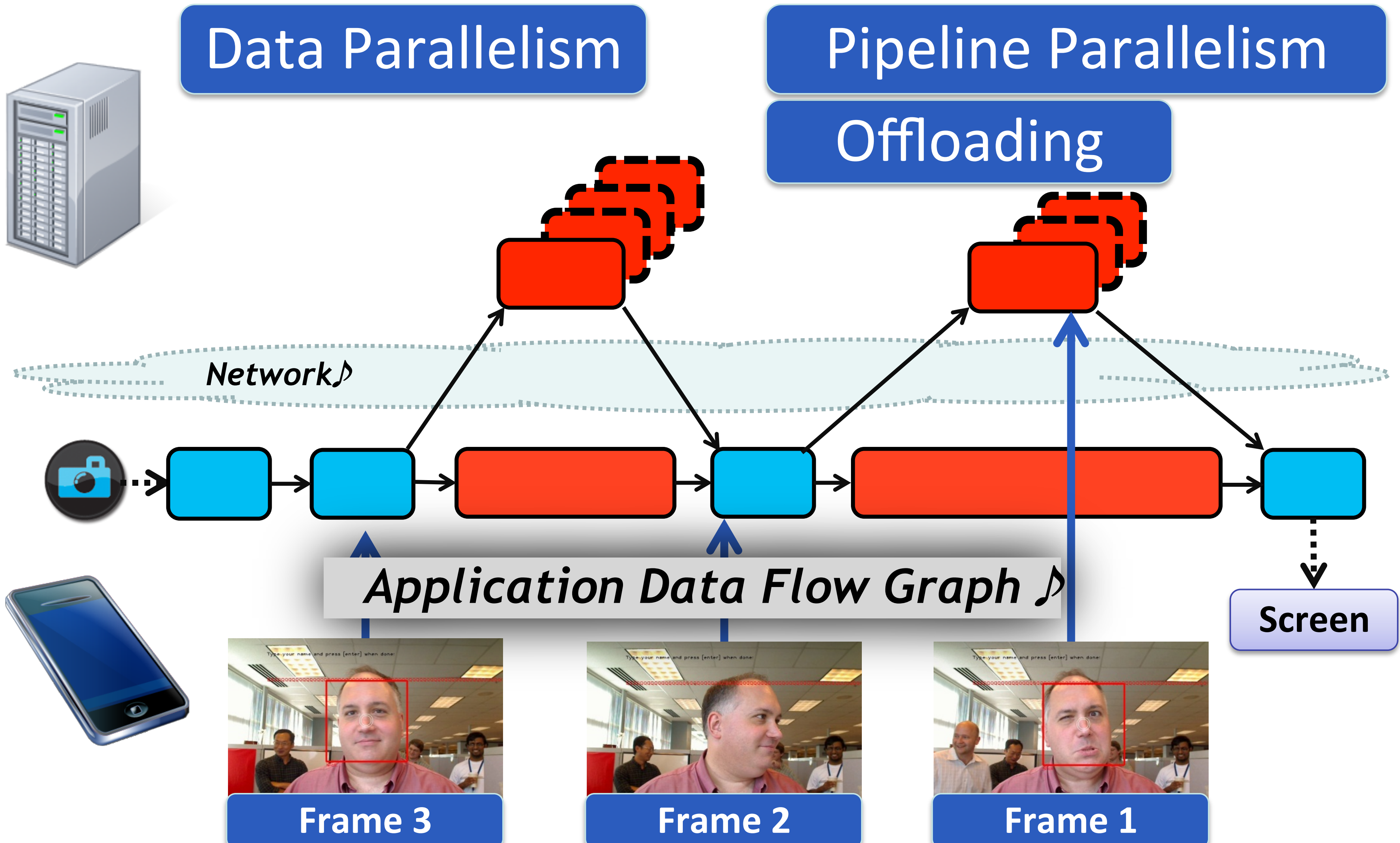
Application	Throughput	Makespan
Face Recognition	2.50 fps	2.09 s
Object and Pose Recognition	0.09 fps	15.8 s
Gesture Recognition	0.42 fps	2.54 s

All running locally on mobile device



Video of 1 fps

Speed-up Techniques



Main Focus

Data Flow Structure

Offloading + Parallelism

System Support

Enable Mobile Interactive Perception Application

Contributions

What factors impact offloading and parallelism?

Measurement

How do we improve throughput and makespan simultaneously?

Odessa Design

How much benefits can we get?

Evaluation

Measurement

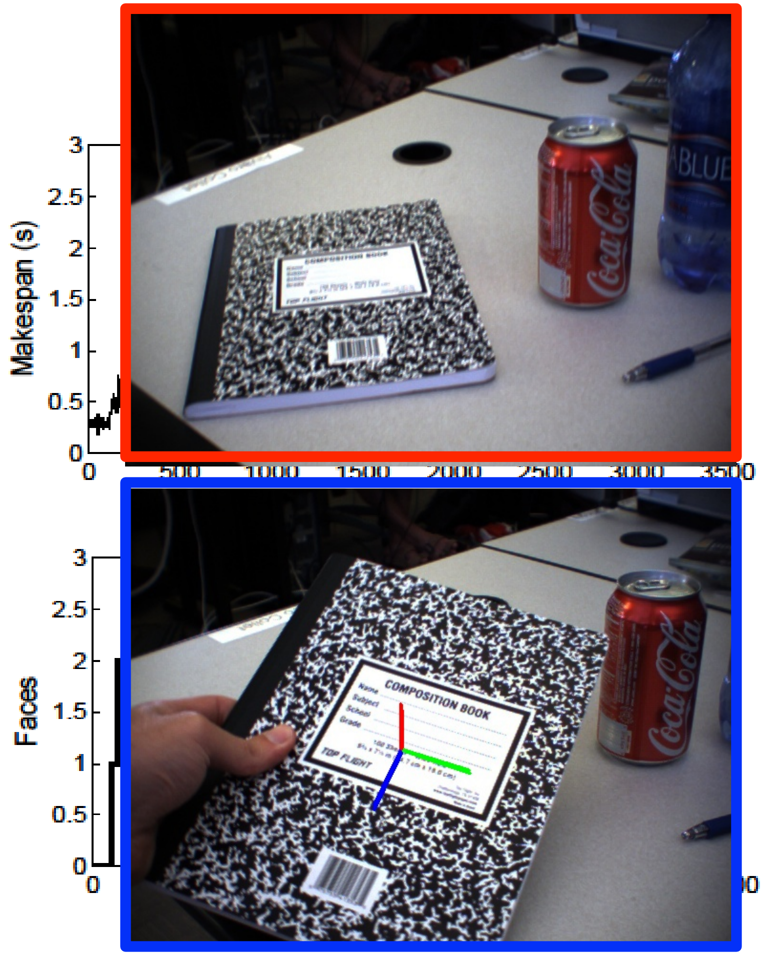
Input Data Variability

Varying Capabilities of Mobile Platform

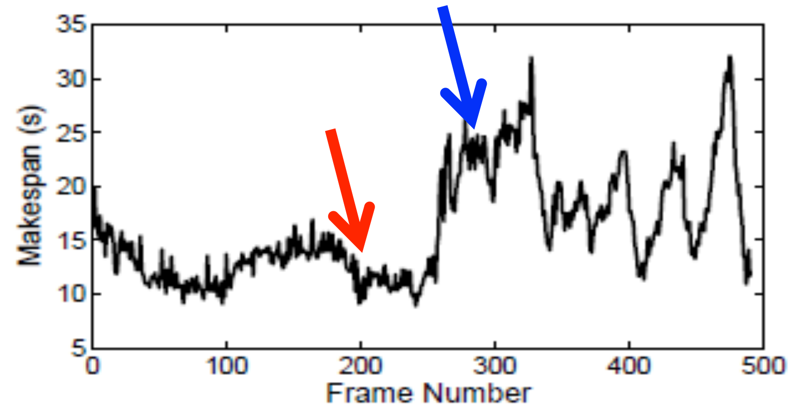
Network Performance

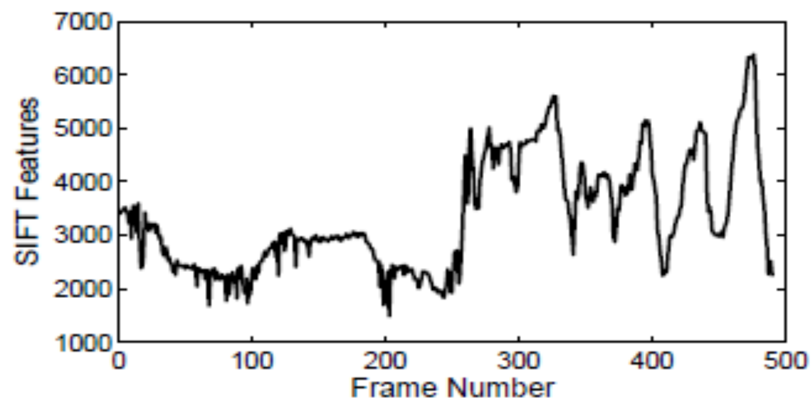
Effects of Parallelism

Lesson I : Input Variability



Object and Pose Recognition



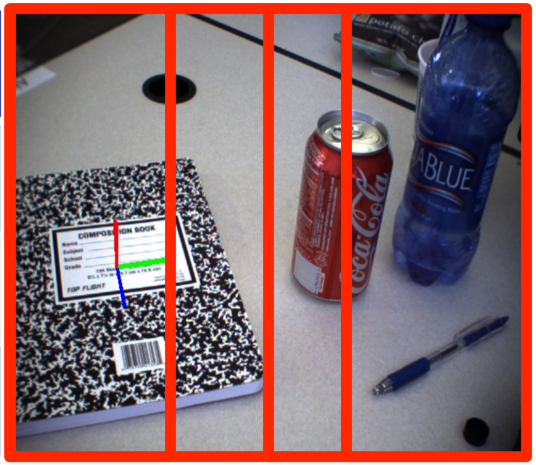


The system should adapt to the variability at runtime

Lesson II: Effects of Data Parallelism

Object and Pose Recognition

# of Threads	Thread 1	Thread 2	Thread 3
1	1,203 ms	-	-
2	741 ms	465 ms	-
3	443 ms	505 ms	233 ms



Input Complexity

Segmentation Method

The level of data parallelism affects accuracy and performance.

Summary: Major Lessons

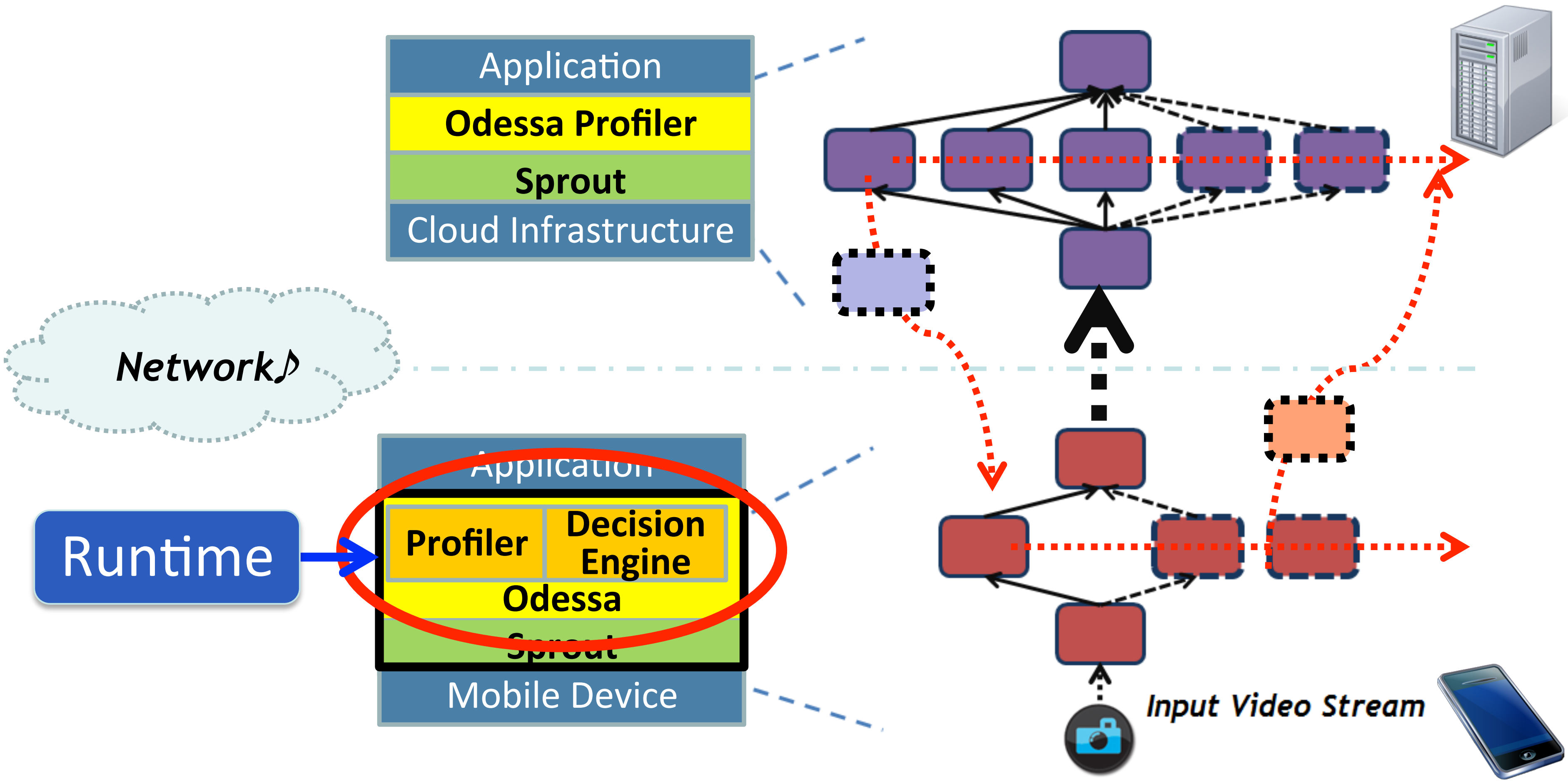
Offloading decisions must be made in an adaptive way.

The level of data parallelism cannot be determined a priori.

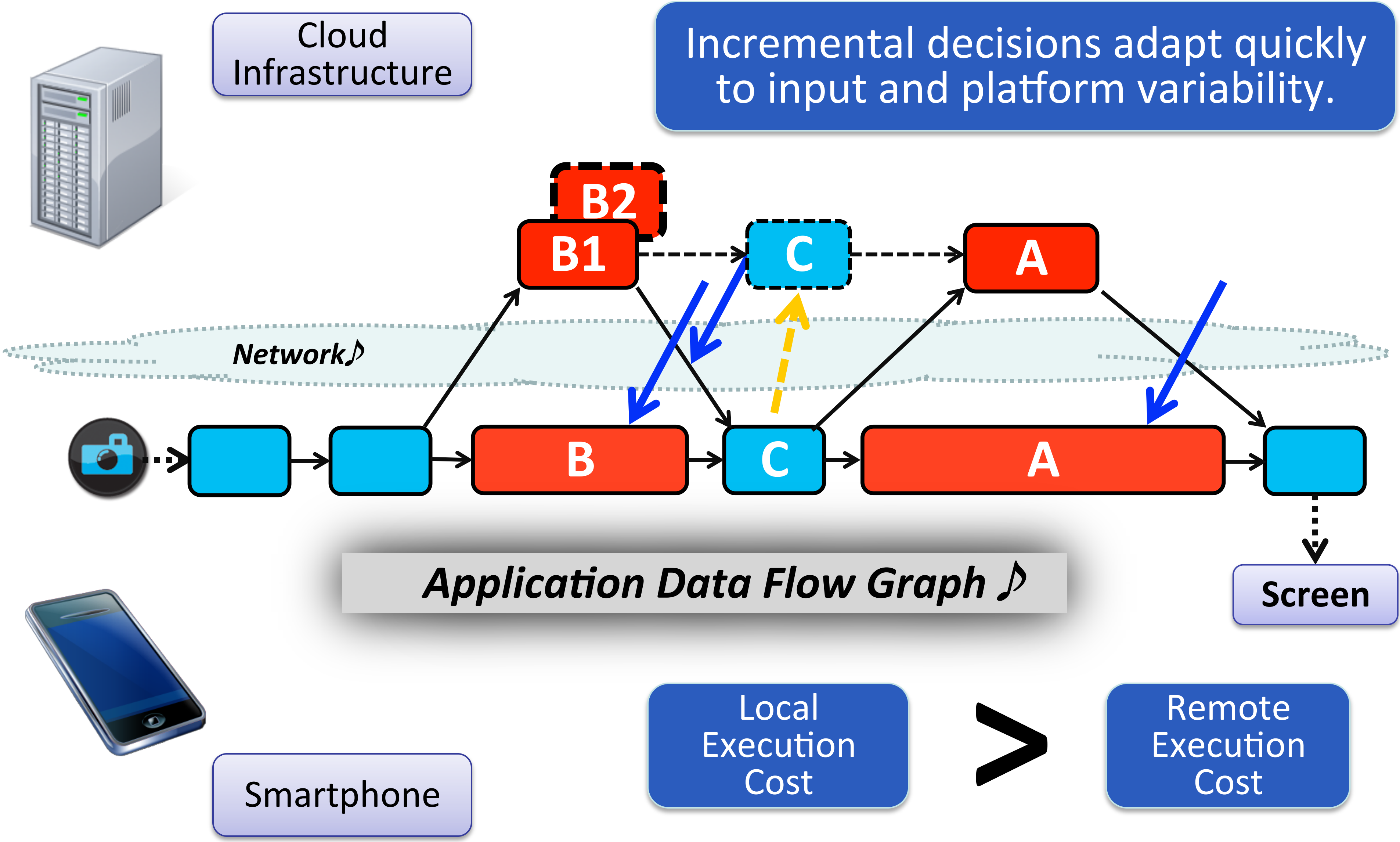
A static choice of pipeline parallelism can cause sub-optimal performance.

Odessa

Offloading DEcision System for Streaming Applications



Incremental Decision Making Process



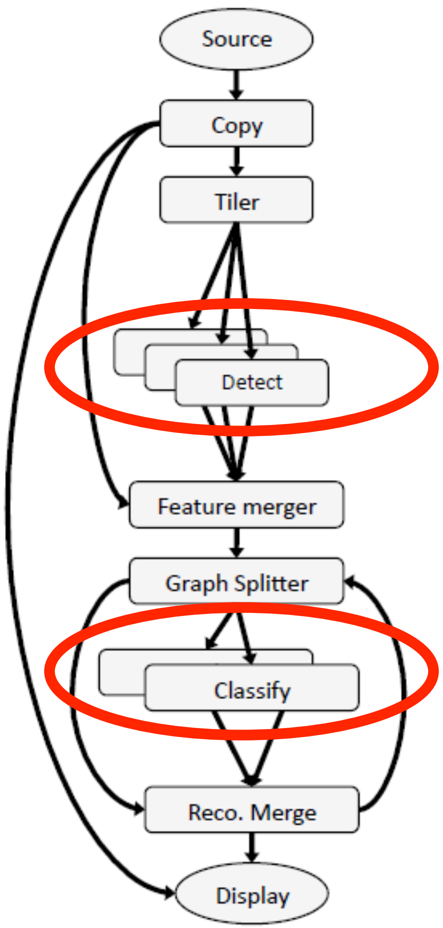
Evaluation Methodology

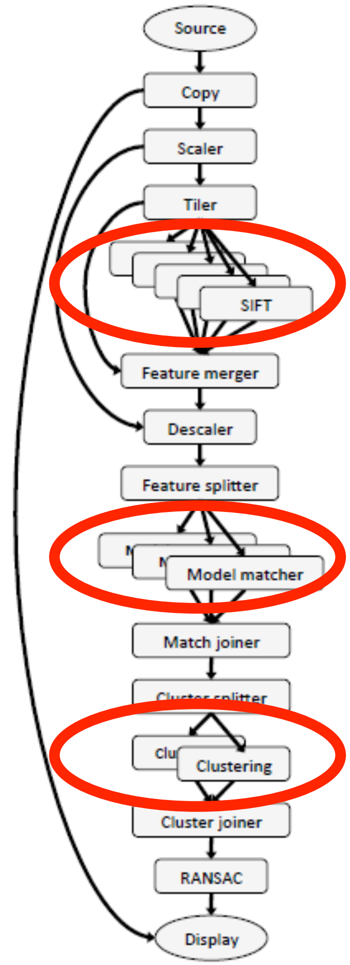
Implementation ← **Linux / C++**

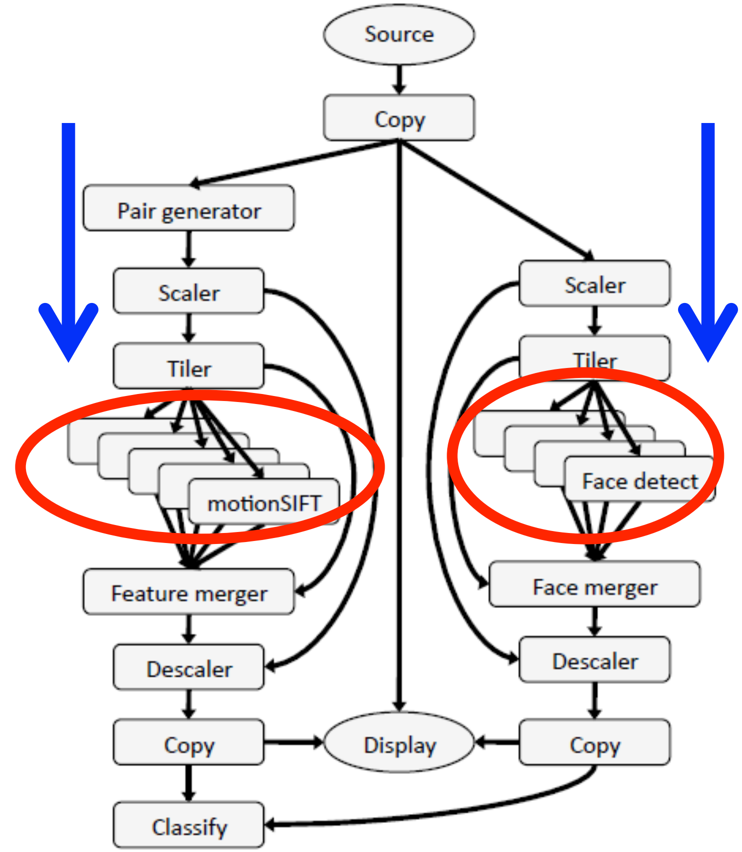
Experiments ← **1-core Netbook**, **2-core Laptop**, **8-core Server**

- **Odessa Adaptation**
- **Resulting Partitions**
- **Performance Comparison**

Data-Flow Graph



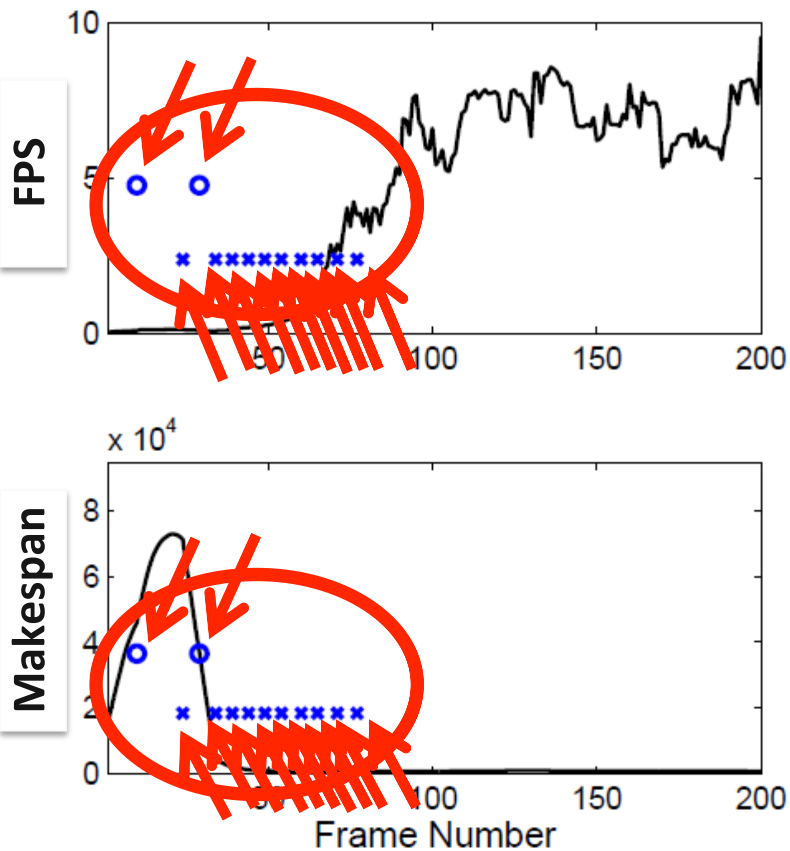




Face Recognition **Object Pose Estimation** **Gesture Recognition**

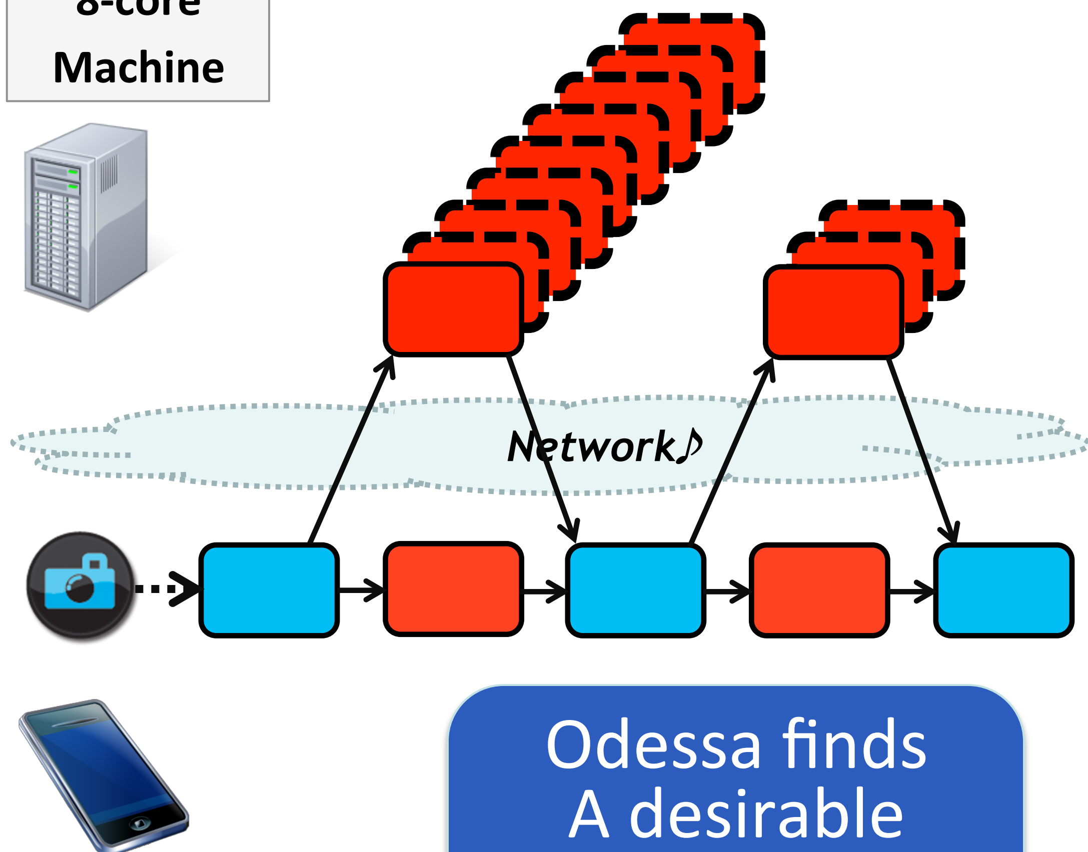
Odessa Adaptation

Object and Pose Recognition



Mobile Device

8-core Machine



1-core

Odessa finds
A desirable
configuration
automatically.

Resulting Partitions in Different Devices

Face Recognition

Client Device	Stage Offloaded and Instances	Degree of Pipeline Parallelism
Mobile Device	Face detection (2)	3.39
Dual Core Notebook	Nothing	3.99

Gesture Recognition

Client Device	Stage Offloaded and Instances	Degree of Pipeline Parallelism
Mobile Device	Face Detection (1) Motion-SIFT Feature (4)	3.06
Dual Core Notebook	Face Detection (1) Motion-SIFT Feature (9)	5.14

Resulting partitions are often very different for different client devices.

Performance Comparison with Other Strategy

Object and Pose Recognition Application

Strategy	Throughput (FPS)	Makespan (Latency)
Local	0.09	15,800 ms
Offload-All	0.76	4,430 ms
Domain-Specific	1.51	2,230 ms
Offline-Optimal	6.49	430 ms
Odessa	**6.27**	**807 ms**

Odessa performs 4x better than
the partition suggested by domain expert,
close to the offline optimal strategy.

Related Work

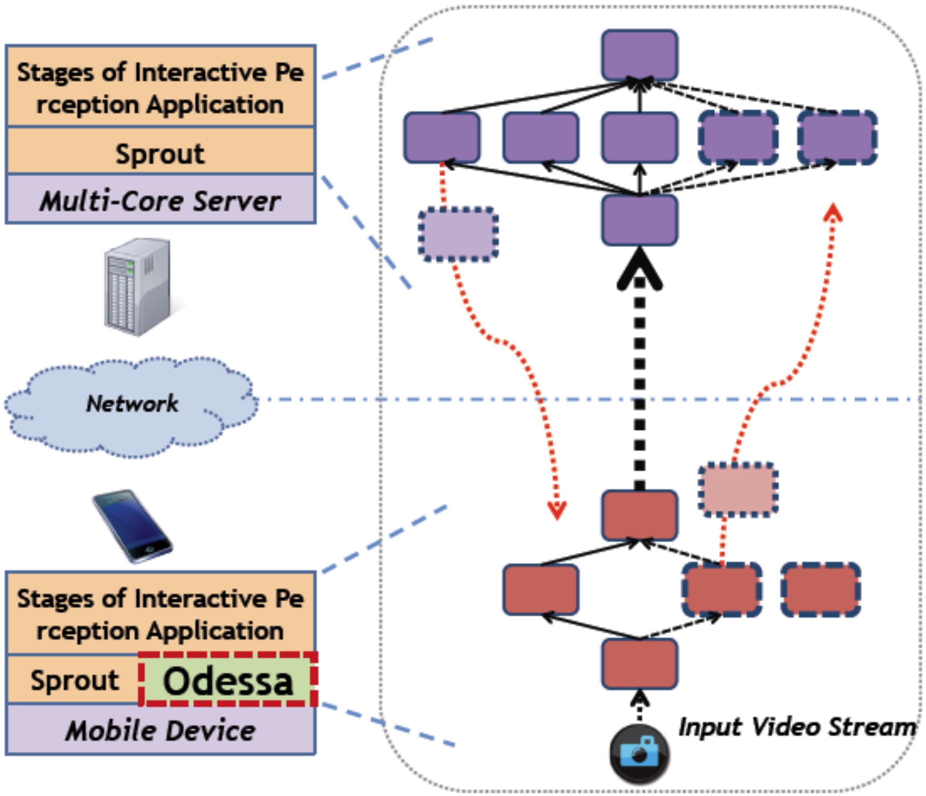
- ***ILP solver*** for saving energy: [MAUI] [CloneCloud]
- ***Graph-based*** partitioning: [Gu'04] [Li'02] [Pillai'09] [Coign]
- ***Static Partitioning***: [Wishbone] [Coign]
- A set of ***pre-specified*** partitions: [CloneCloud] [Chroma] [Spectra]

Objectives | Variability | Migration, Contention | Parallelization

↓

Odessa

Summary of Odessa



Adaptive & Incremental runtime for mobile perception applications

- Odessa system design using novel workloads.
- Understanding of the factors which contribute to the offloading and par allelism decisions.
- Extensive evaluation on prototype implementation.

Odessa’s quick adaptation?

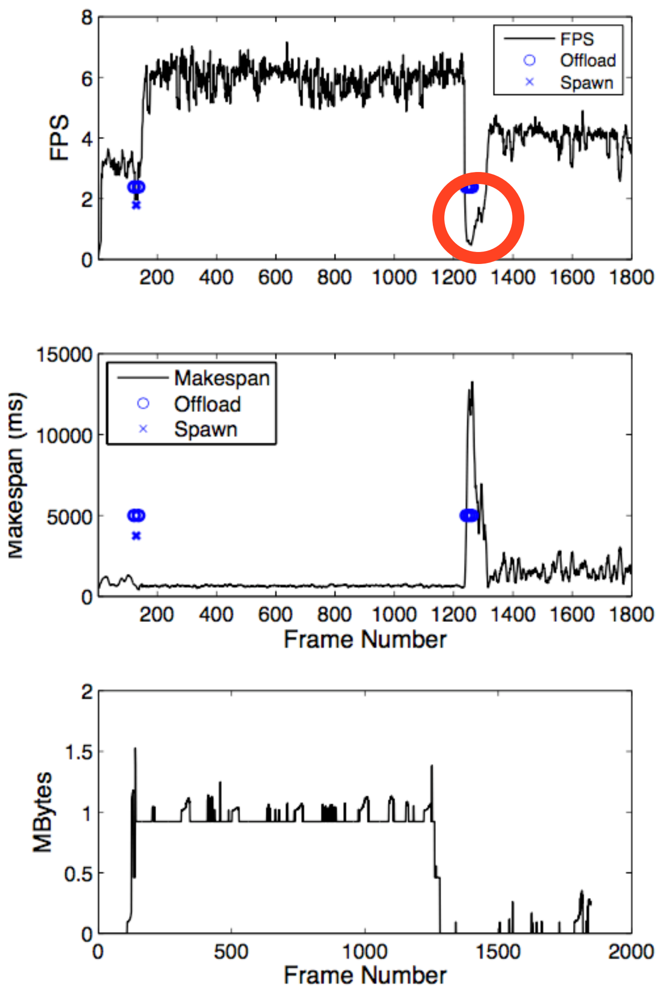


Figure 14: *Odessa* adapting to changes in network performance. The network bandwidth is reduced from 100 Mbps to 5 Mbps at frame number 1237. *Odessa* pulls back the offloaded stages from the server to the local machine to reduce the data transmitted over the network.

- It takes 70 frames to adapt to new network condition.
- Throughput during that period: ~1.5fps
- So it took almost 47 seconds to adapt!

Other Thoughts

- Limited to stream processing apps (mostly because of Sprout)
- The security risks totally ignored
- The implementation not built for cloud
- Adding content-aware data parallelism to improve app fidelity loss
- They could also present the power gains.

Thank you

“Any questions?”