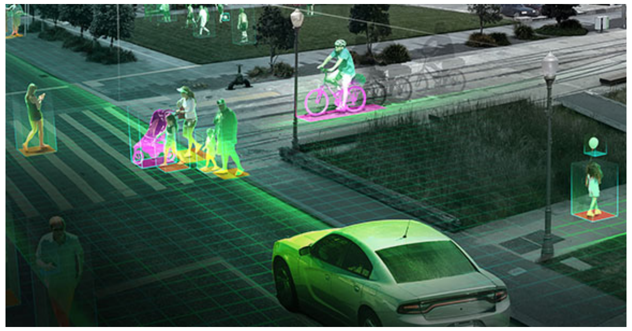
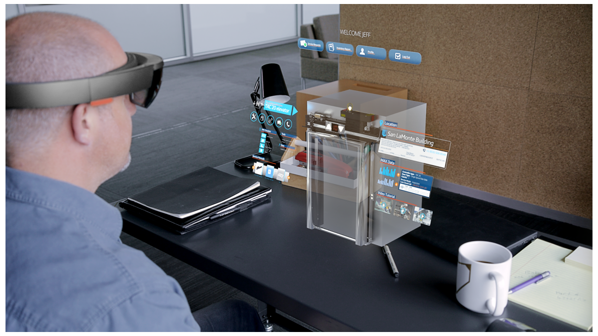
LAVEA: Latency-Aware Video Analytics on Edge Computing Platform

Shanhe Yi[*], Zijiang Hao[*], Qingyang Zhang[⌈⌇], Quan Zhang[⌈], Weisong Shi[⌈], Qun Li[*]
College of William and Mary[*]
Wayne State University[⌈]
Anhui University, China[⌇]

Video Data is a Gold Mine



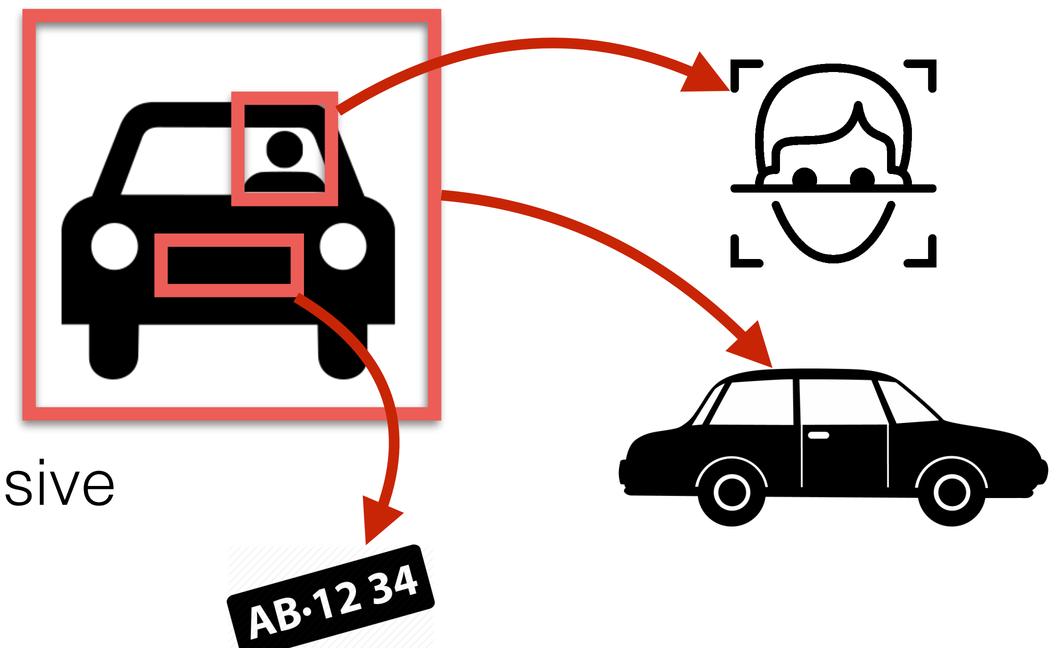


You have to do it quickly.

How to do *low-latency* video analytics?

William & Mary
CHARTERED 1693

Motivation - Amber Alert

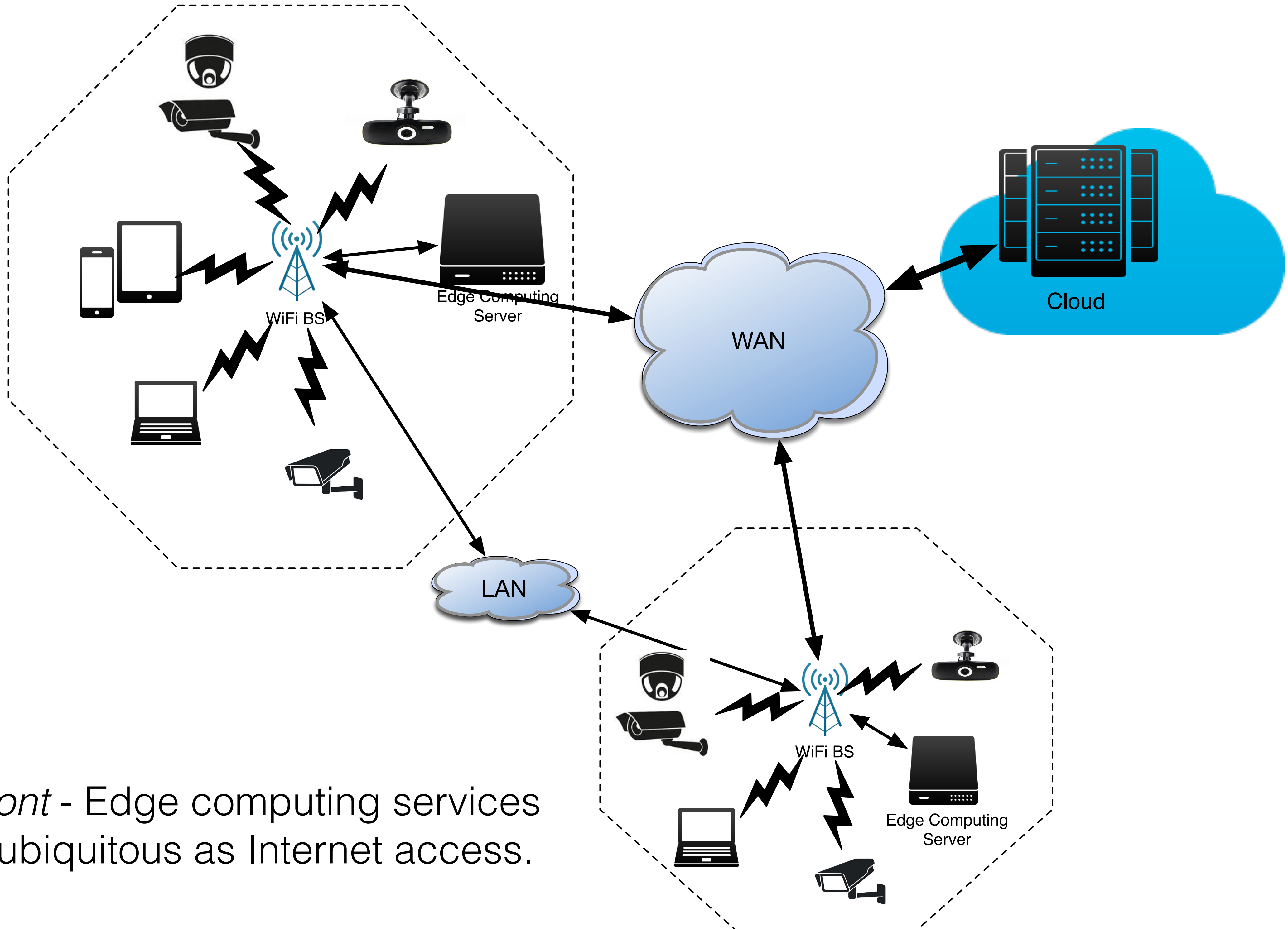


- Video analytic tasks are computation-intensive and bandwidth-hungry
- Run on mobile or IoT devices
  - Computational latency
  - Battery drain
  - Heat dissipation
- Run on cloud:
  - Transmission latency
  - Bandwidth cost

- Image acquisition
- License plate extraction
- License plate analysis
- Character recognition

How Edge Computing can Help?

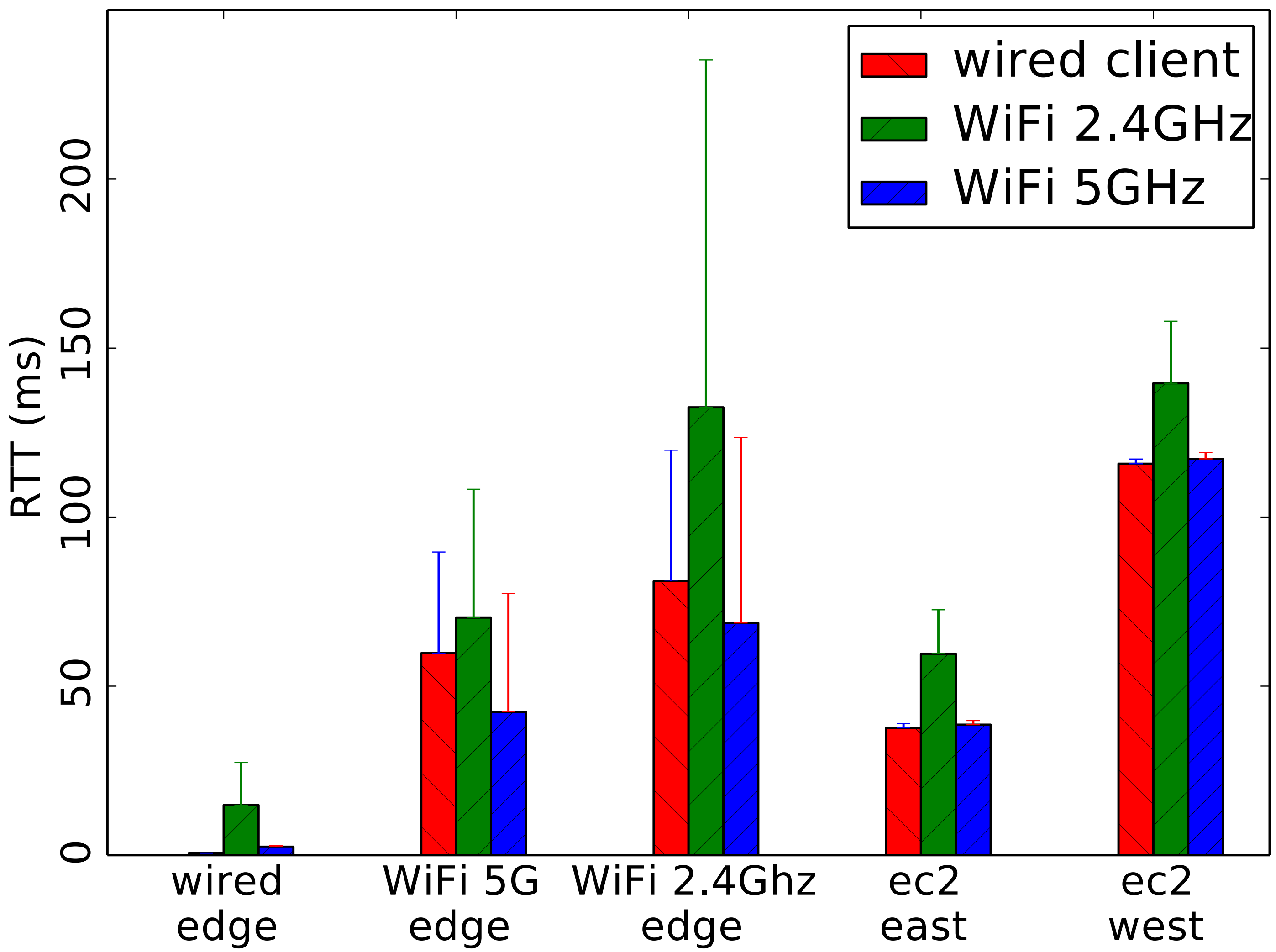
- Edge Computing Network

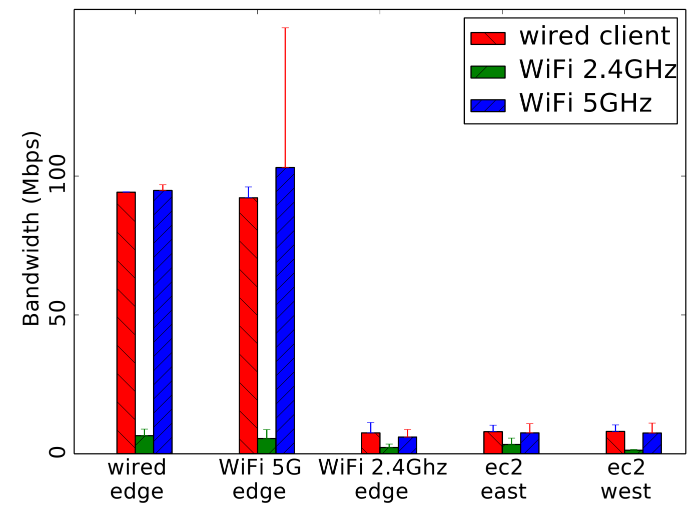


Edge-front - Edge computing services can be ubiquitous as Internet access.

How Edge Computing can Help?

- Feasibility of leveraging edge computing node

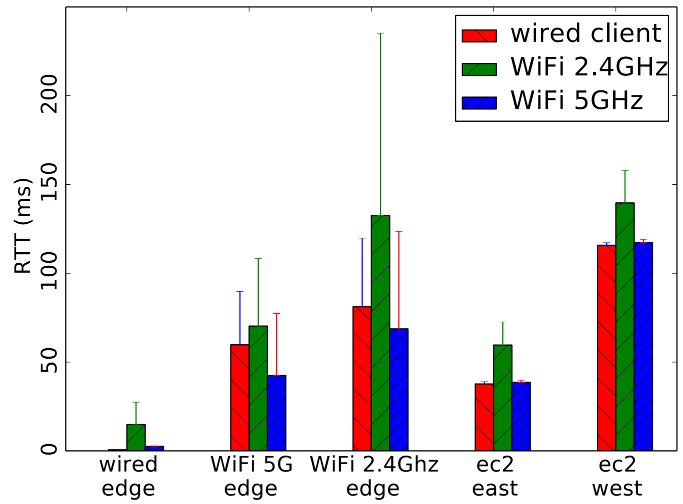


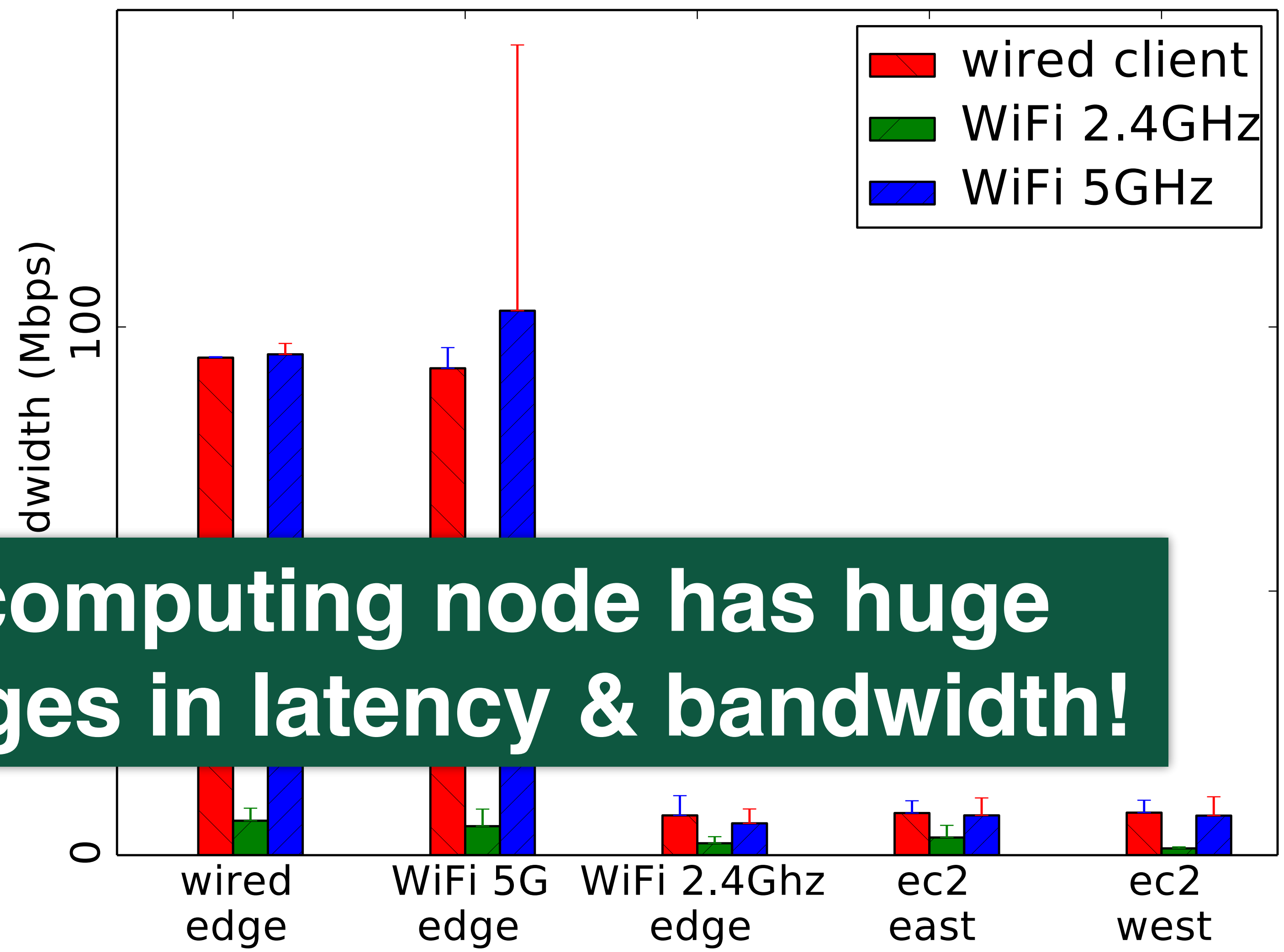


Round-trip time (RTT) : Wired connection is the best; WiFi 5GHz has larger mean and variance compared to the cloud node in the the closest region;

How Edge Computing can Help?

- Feasibility of leveraging edge computing node





Edge computing node has huge advantages in latency & bandwidth!

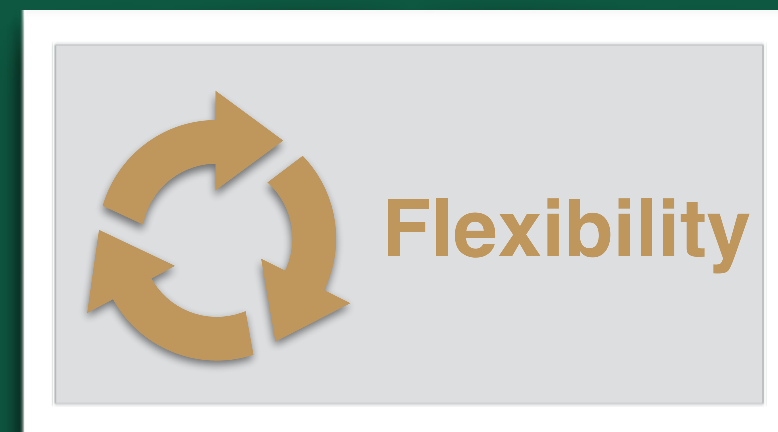
Bandwidth (BW) : All clients have benefits in utilizing a wired or advanced-wireless edge computing node.

How shall we provide low-latency video analytics in edge computing system?

Video Analytics meets Edge Computing

Response Time Minimization Problem

- Client task offloading selecting
- Offloaded task prioritizing
- Offloaded task placing

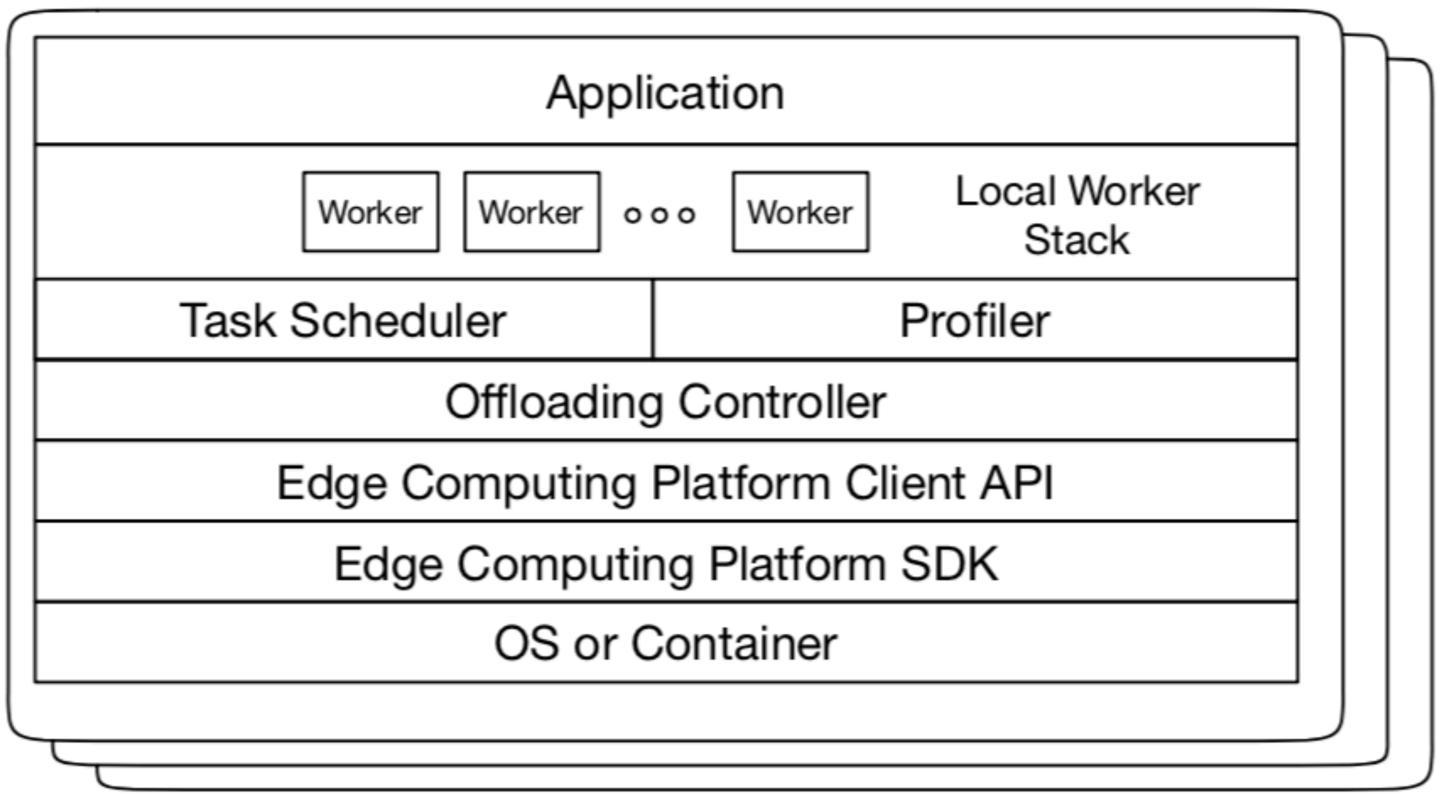


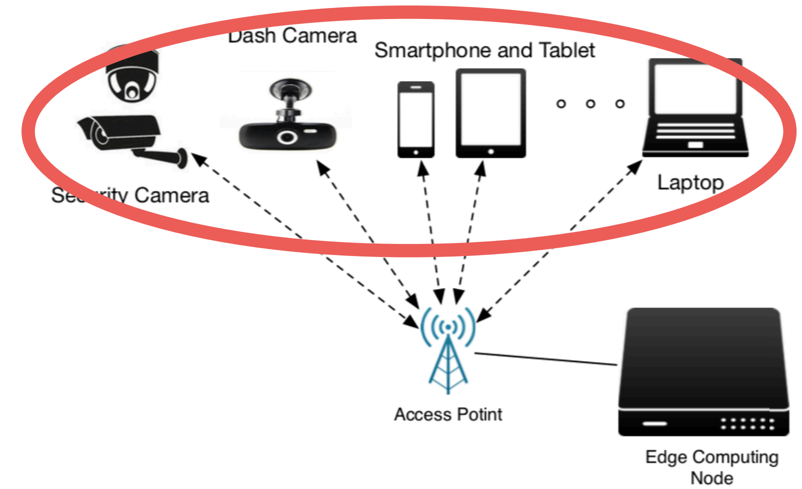
Edge Computing Platform Design

- Serverless architecture
- Edge computing service
  - Offloading service
  - Queueing service
  - Scheduling service

- Introduction
- **System Design Overview**
- Edge Computing Services
- Evaluation
- Conclusion

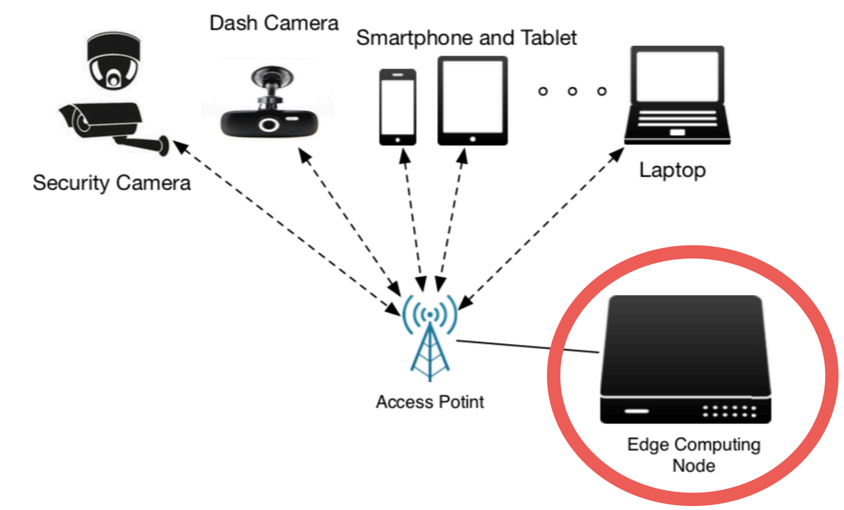
System Design - Edge Client





- Resource-constrained devices
    - Run lightweight data processing locally
    - Offload heavy tasks to nearby edge computing nodes
- **Profiler**
    - Collect task performance
- Offloading Controller
    - Act as an agent to fulfill offloading decisions

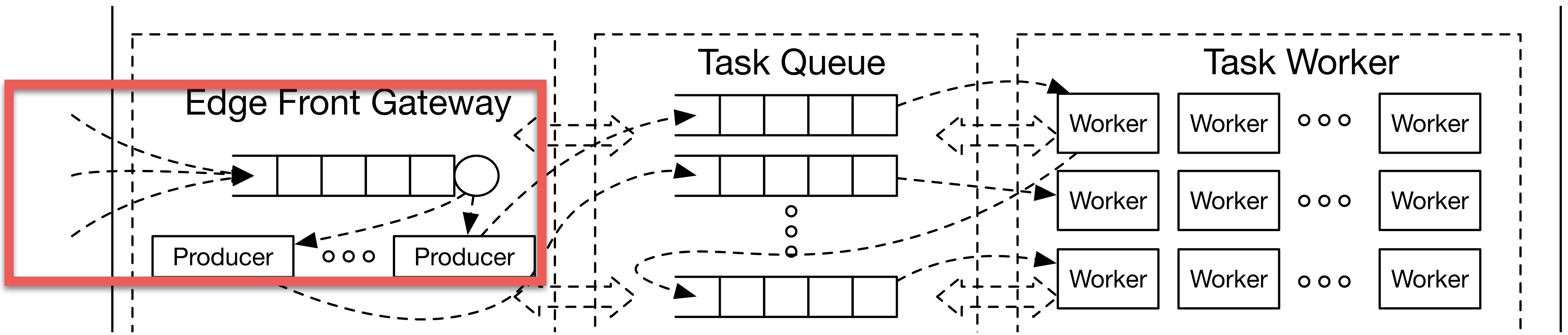
System Design - Edge Computing Node



Host OS

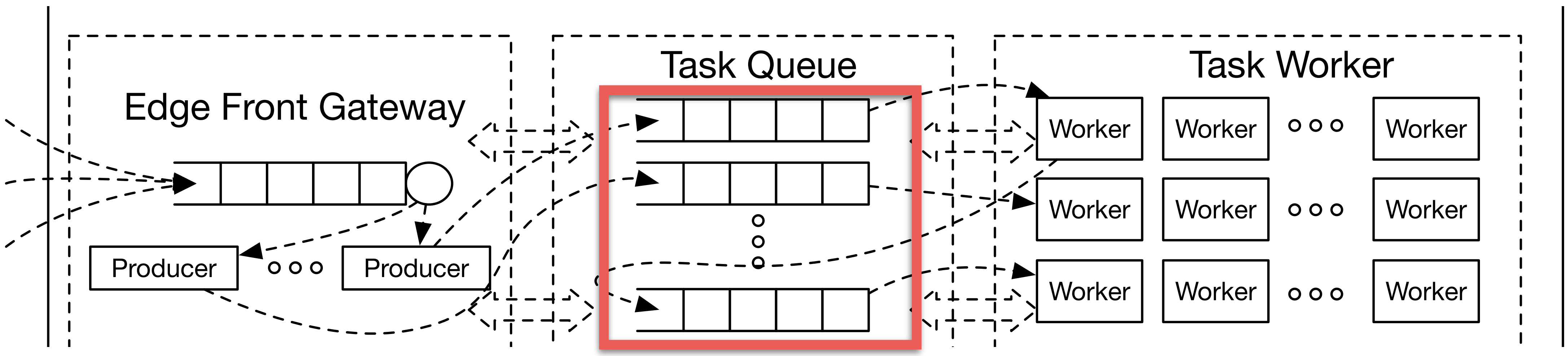
- Docker container - resource allocation/isolation, easy deployment
- Modular services
- Serverless architecture (Function-as-a-Service)
  - AWS Lambda@Edge, Apache OpenWhisk
  - Event-based micro-service framework

System Design - Edge Computing Node



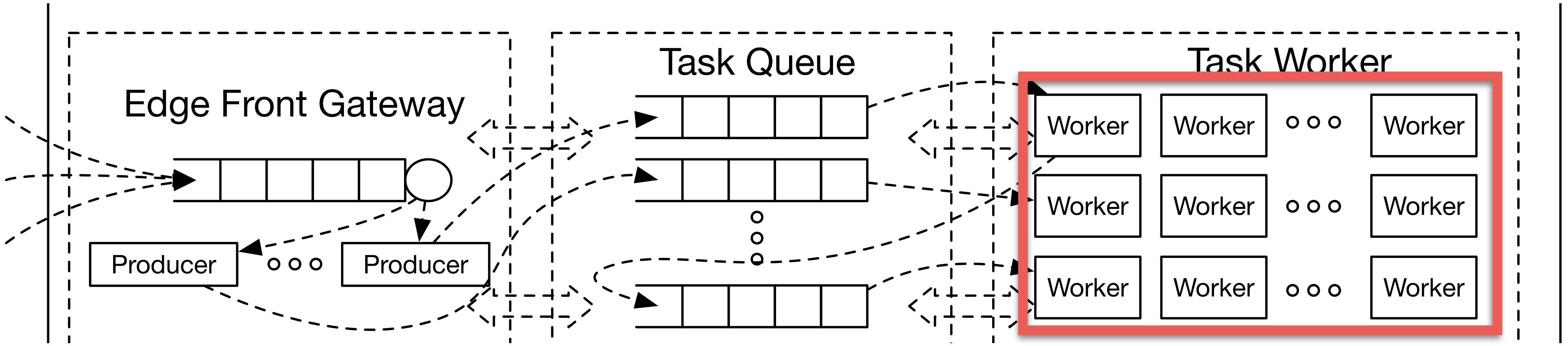
- user request
  - event of interests (e.g. **plate**, face, car)
  - input source
  - the event handler code (function), build the docker image
  - execution configuration (e.g. cron job, where to save the result, or trigger another event)
  - resource configuration (limit container resource)

System Design - Edge Computing Node



- user request -> task (in a format docker command along with input)
  - event of interests (e.g. **plate**, face, car)
  - input source
  - the event handler code (function) -> docker image
  - a script for docker to run

System Design - Edge Computing Node

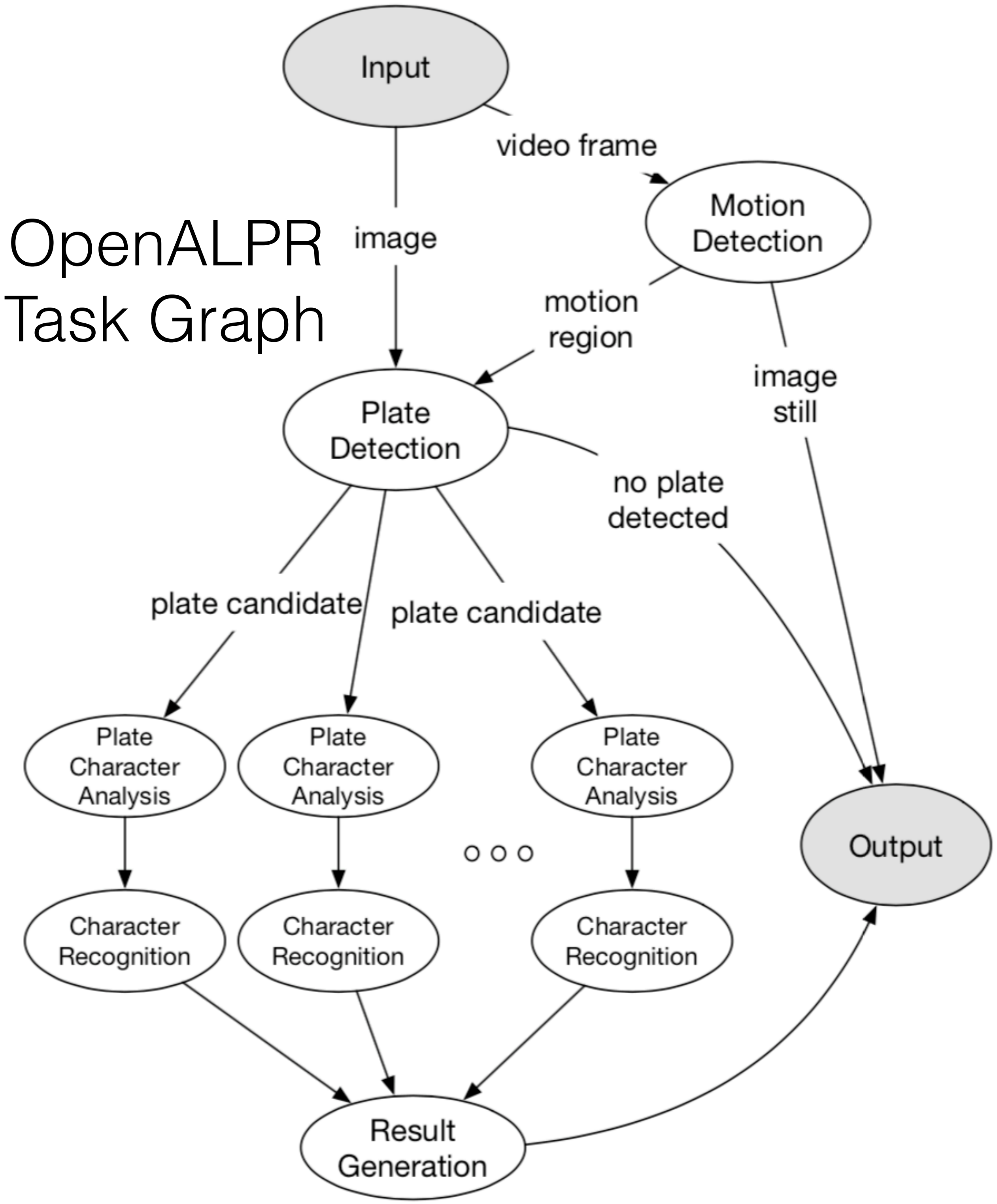


- consume the tasks in docker container instances

- Introduction
- System Design Overview
- **Edge Computing Services**
  - **Offloading Service - Client Task Offloading Problem**
  - **Queueing Service - Offloaded Task Prioritizing**
  - **Scheduling Service - Offloaded Task Placement**
- Evaluation
- Conclusion

Client Task Offloading - System Model

OpenALPR Task Graph



Directed Acyclic Graph

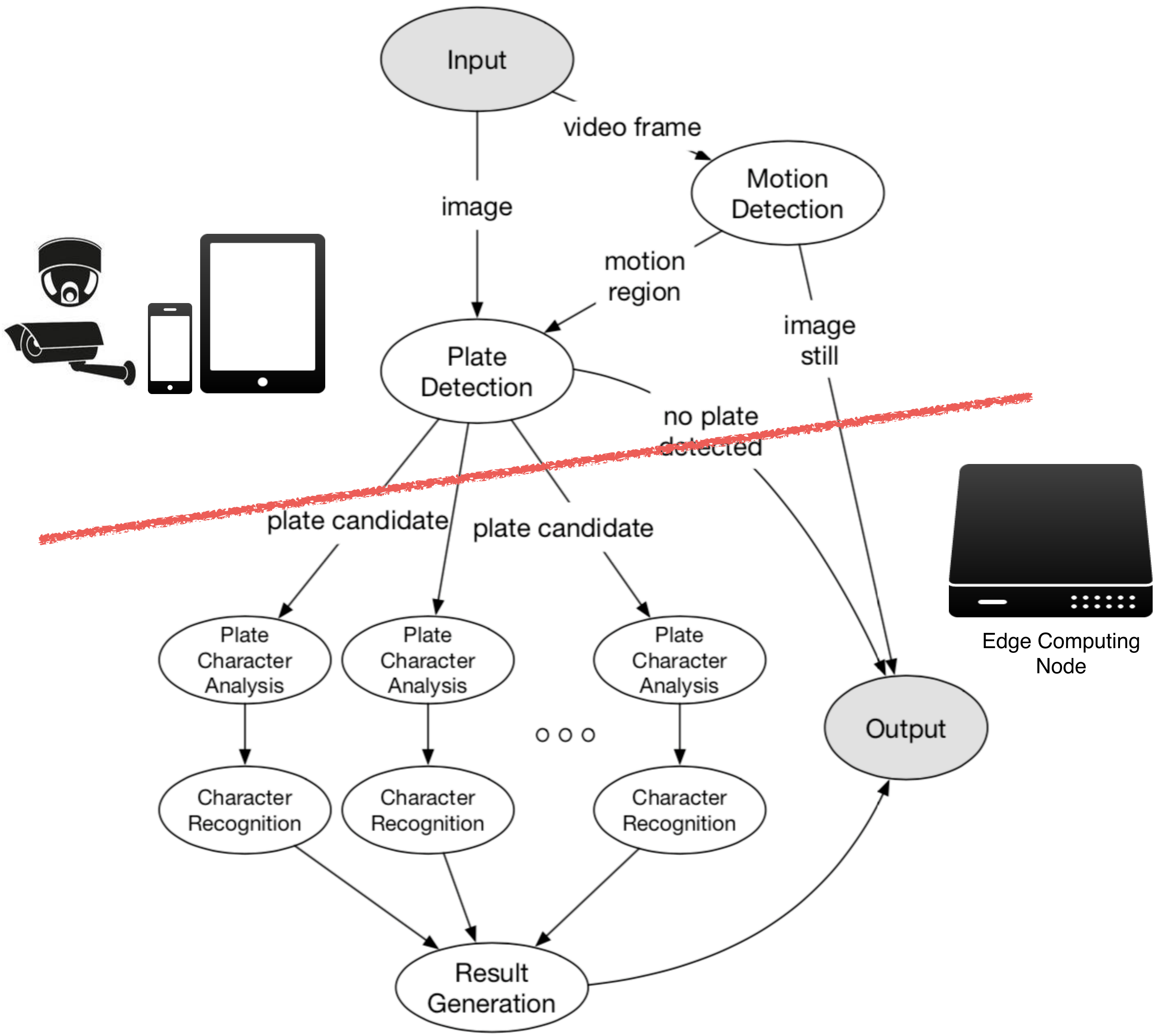
$$G = (V, E)$$

Each vertex $v \in V$ weight is the computation cost of a task (c_v)

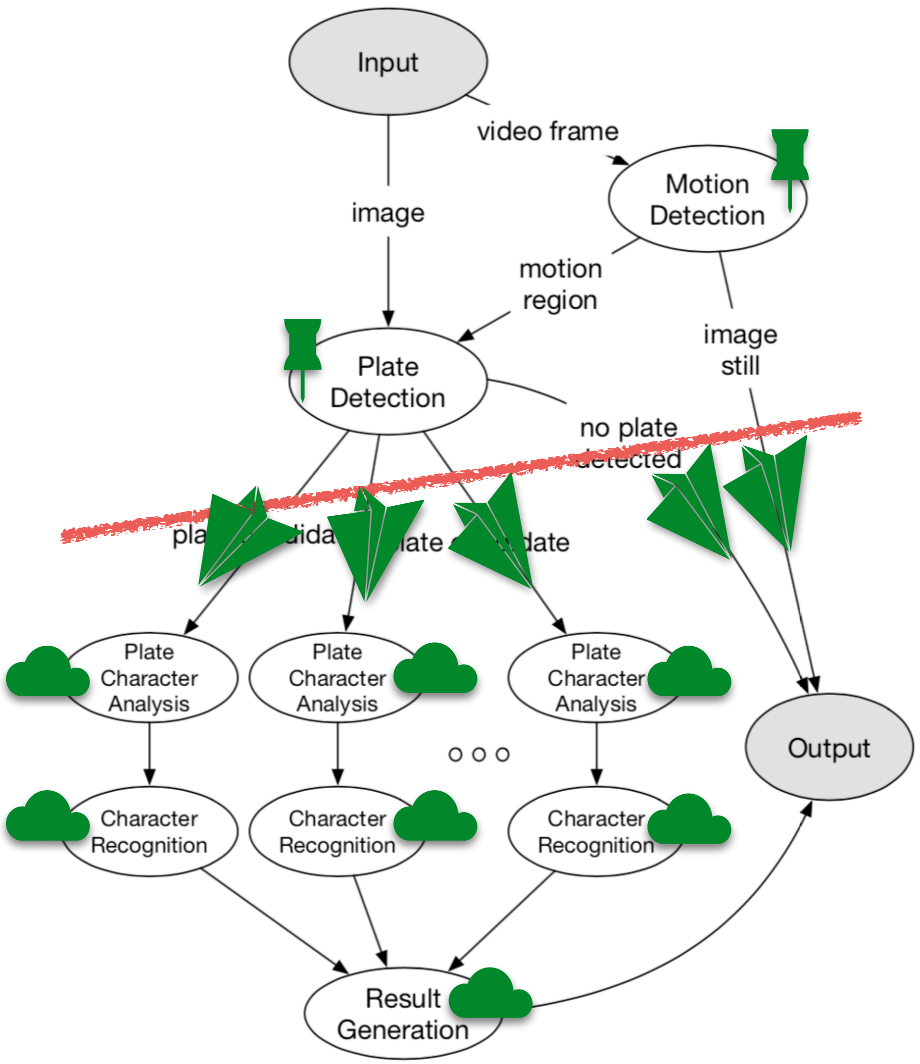
Each edge $e = (u, v), u, v \in V$ weight is the data size of intermediate result $(d_{u,v})$

Weights are gathered via profilers, as pre-runtime information.

Client Task Offloading - Problem Formulation



Client Task Offloading - Problem Formulation



$$\min_{\mathbf{I}_i, r_i} \sum_{i=1}^{N} (T_i^{local} + T_i^{net} + T_i^{remote})$$

The loc
of client

T_i^{local}

c_v the co
p_i the pr

The remote execution time

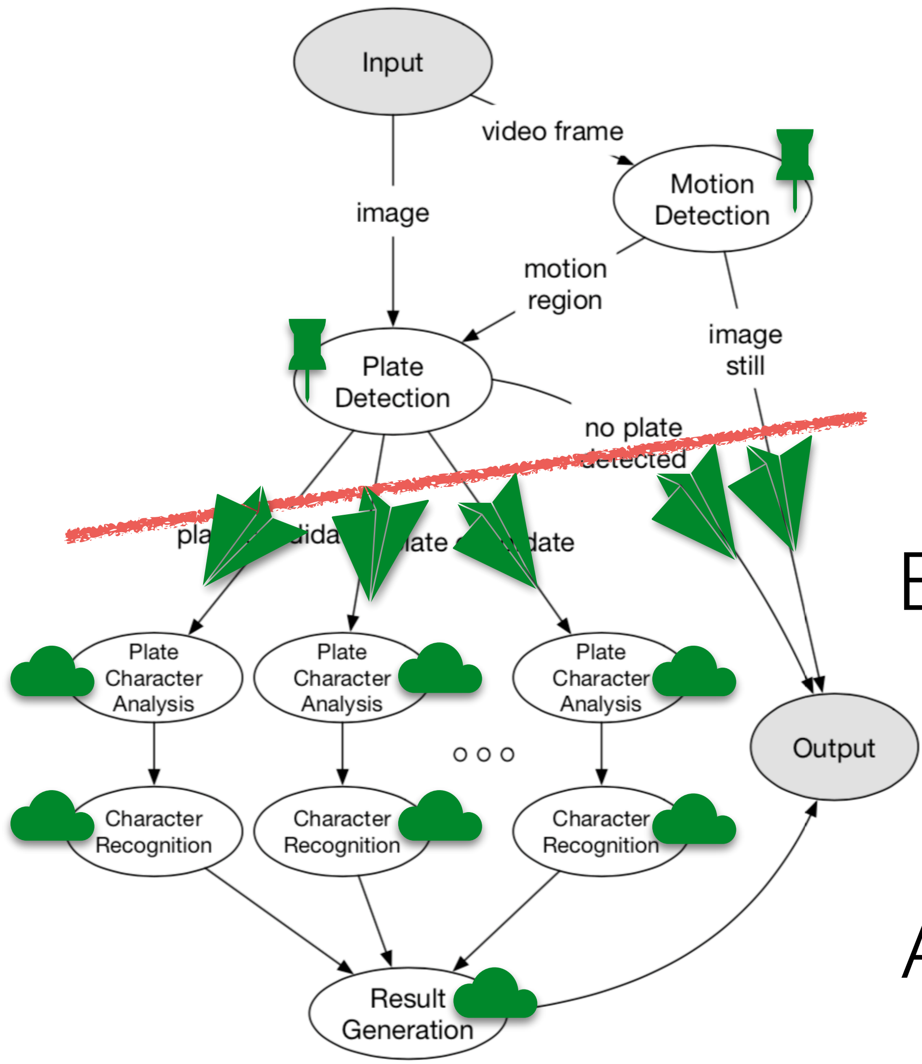
$$T_i^{remote} = \sum_{v \in V} (1 - I_{v,i})(c_v / p_0)$$

c_v the computation cost
p_0 the edge processor speed

For each client $i, i \in [1, N]$
We use an indicator $I_{v,i} \in \{0, 1\}$
If $I_{v,i} = 1$, task v at client i runs loc
Otherwise, run remotely

WILLIAM & MARY
CHARTERED 1693

Client Task Offloading - Problem Formulation



$$\min_{\mathbf{I}_i, r_i} \sum_{i=1}^{N} (T_i^{local} + T_i^{net} + T_i^{remote})$$

Bandwidth constraint

$$\text{s.t.} \quad \sum_{i=1}^{N} r_i \le R$$

Avoid ping-pong constraint

$$\text{s.t.} \quad I_{v,i} \le I_{u,i}, \forall e(u,v) \in E, \forall i \in [1, N]$$

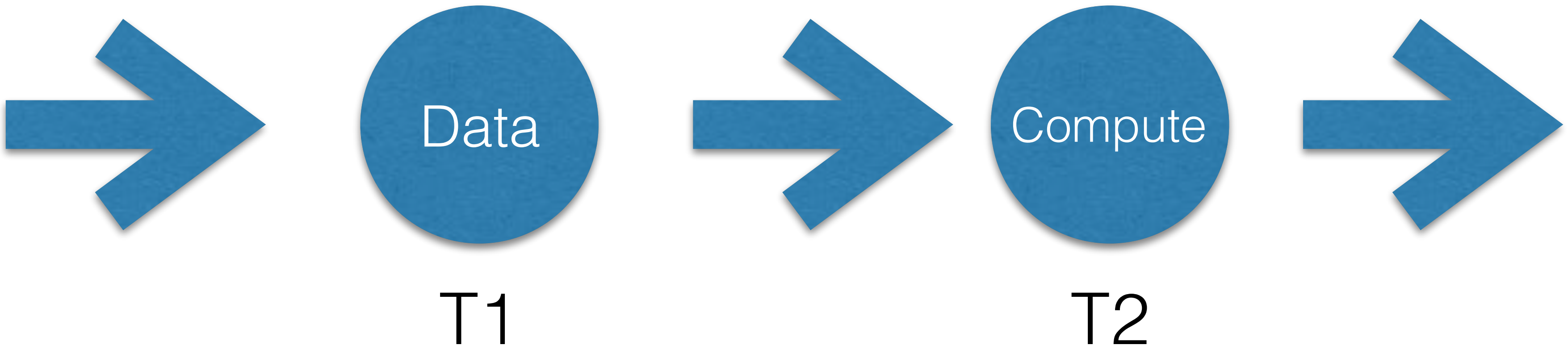
Delay tolerate constraint

$$\text{s.t.} \quad \overline{T}_i^{local} - (T_i^{net} + T_i^{remote}) > \tau, \forall i \in [1, N]$$

Mixed Integer Non-Linear Programming (MINLP)

- relax the integer constrains
- solve the NLP using a constrained nonlinear optimization solver (SQP)
- branch & bound
  - brutal force

Prioritizing Edge Task Queue



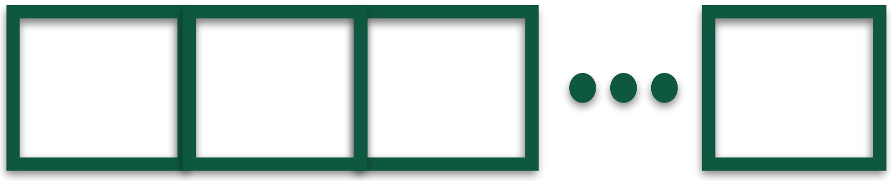
- An offloaded task - two stage
  - Wait for the input or intermediate data (e.g., image or video)
  - Processing the data and return result

Prioritizing Edge Task Queue

- Schedule offloaded tasks to minimize the makespan time
    - Flow job shop model and Johnson's rule [1]

JOB	T1	T2
J1	1	3
J2	5	2
J3	4	6

T1? -> Head



T2? -> Tail

Pick Job with smallest stage time

[1] Selmer Martin Johnson. 1954. Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1, 1 (1954), 61–68.
[2] KR Baker. 1990. Scheduling groups of jobs in the two-machine ow shop. Math- ematical and Computer Modelling 13, 3 (1990), 29–36.

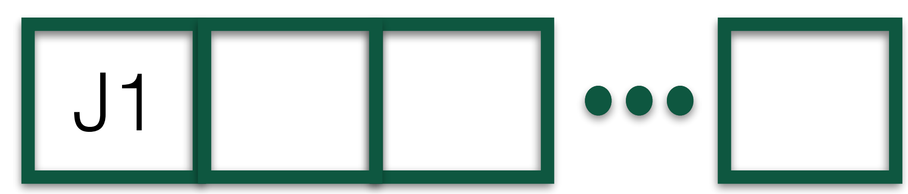
William & Mary
Chartered 1693

Prioritizing Edge Task Queue

- Schedule offloaded tasks to minimize the makespan time
  - Flow job shop model and Johnson's rule [1]

JOB	T1	T2
J1	1	3
J2	5	2
J3	4	6

T1? -> Head



T2? -> Tail

Pick Job with smallest stage time

[1] Selmer Martin Johnson. 1954. Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1, 1 (1954), 61–68.
[2] KR Baker. 1990. Scheduling groups of jobs in the two-machine ow shop. Math- ematical and Computer Modelling 13, 3 (1990), 29–36.

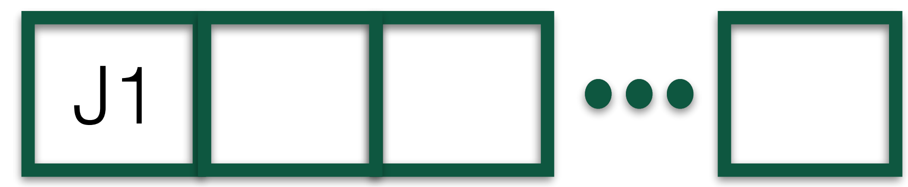
WILLIAM & MARY
CHARTERED 1693

Prioritizing Edge Task Queue

- Schedule offloaded tasks to minimize the makespan time
  - Flow job shop model and Johnson's rule [1]

JOB	T1	T2
~~J1~~	~~1~~	~~3~~
J2	5	2
J3	4	6

T1? -> Head



T2? -> Tail

Pick Job with smallest stage time

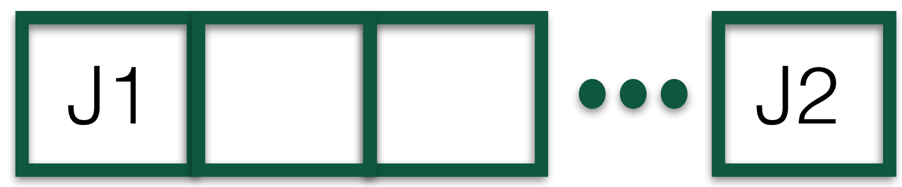
[1] Selmer Martin Johnson. 1954. Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1, 1 (1954), 61–68.
[2] KR Baker. 1990. Scheduling groups of jobs in the two-machine ow shop. Math- ematical and Computer Modelling 13, 3 (1990), 29–36.

Prioritizing Edge Task Queue

- Schedule offloaded tasks to minimize the makespan time
  - Flow job shop model and Johnson's rule [1]

JOB	T1	T2
~~J1~~	~~1~~	~~3~~
J2	5	2
J3	4	6

T1? -> Head



T2? -> Tail

Pick Job with smallest stage time

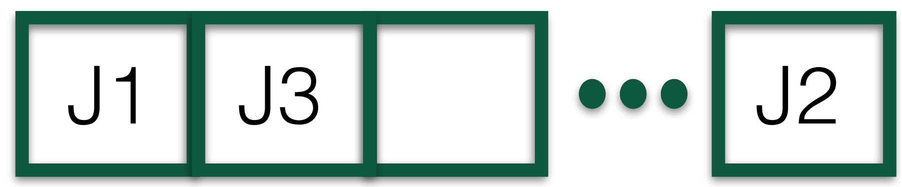
[1] Selmer Martin Johnson. 1954. Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1, 1 (1954), 61–68.
[2] KR Baker. 1990. Scheduling groups of jobs in the two-machine ow shop. Math- ematical and Computer Modelling 13, 3 (1990), 29–36.

Prioritizing Edge Task Queue

- Schedule offloaded tasks to minimize the makespan time
  - Flow job shop model and Johnson's rule [1]

JOB	T1	T2
~~J1~~	~~1~~	~~3~~
~~J2~~	~~5~~	~~2~~
J3	4	6

T1? -> Head



T2? -> Tail

Pick Job with smallest stage time

[1] Selmer Martin Johnson. 1954. Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1, 1 (1954), 61–68.
[2] KR Baker. 1990. Scheduling groups of jobs in the two-machine ow shop. Math- ematical and Computer Modelling 13, 3 (1990), 29–36.

Prioritizing Edge Task Queue

- Schedule offloaded tasks to minimize the makespan time
  - Flow job shop model and Johnson's rule [1]
    - how to apply when there are dependencies?

  - **Our heuristic solution for tasks with dependencies [2]**
    - Group tasks with dependencies
    - In each group, find the topological order with minimal makespan time
    - Apply Johnson's rule directly on task groups

[1] Selmer Martin Johnson. 1954. Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1, 1 (1954), 61–68.
[2] KR Baker. 1990. Scheduling groups of jobs in the two-machine ow shop. Math- ematical and Computer Modelling 13, 3 (1990), 29–36.

Inter Edge Collaboration

- Motivation
  - With increasing number of client node nearby, edge-front node can be overloaded and non-responsive to new requests
  - Collaborate with nearby edge node by placing tasks to some not-so-busy neighbor edge nodes

- Problem
  - Given an edge-front node and its neighbors, when the edge-front is overloaded, how to select neighbor as task placement target?

Inter Edge Collaboration

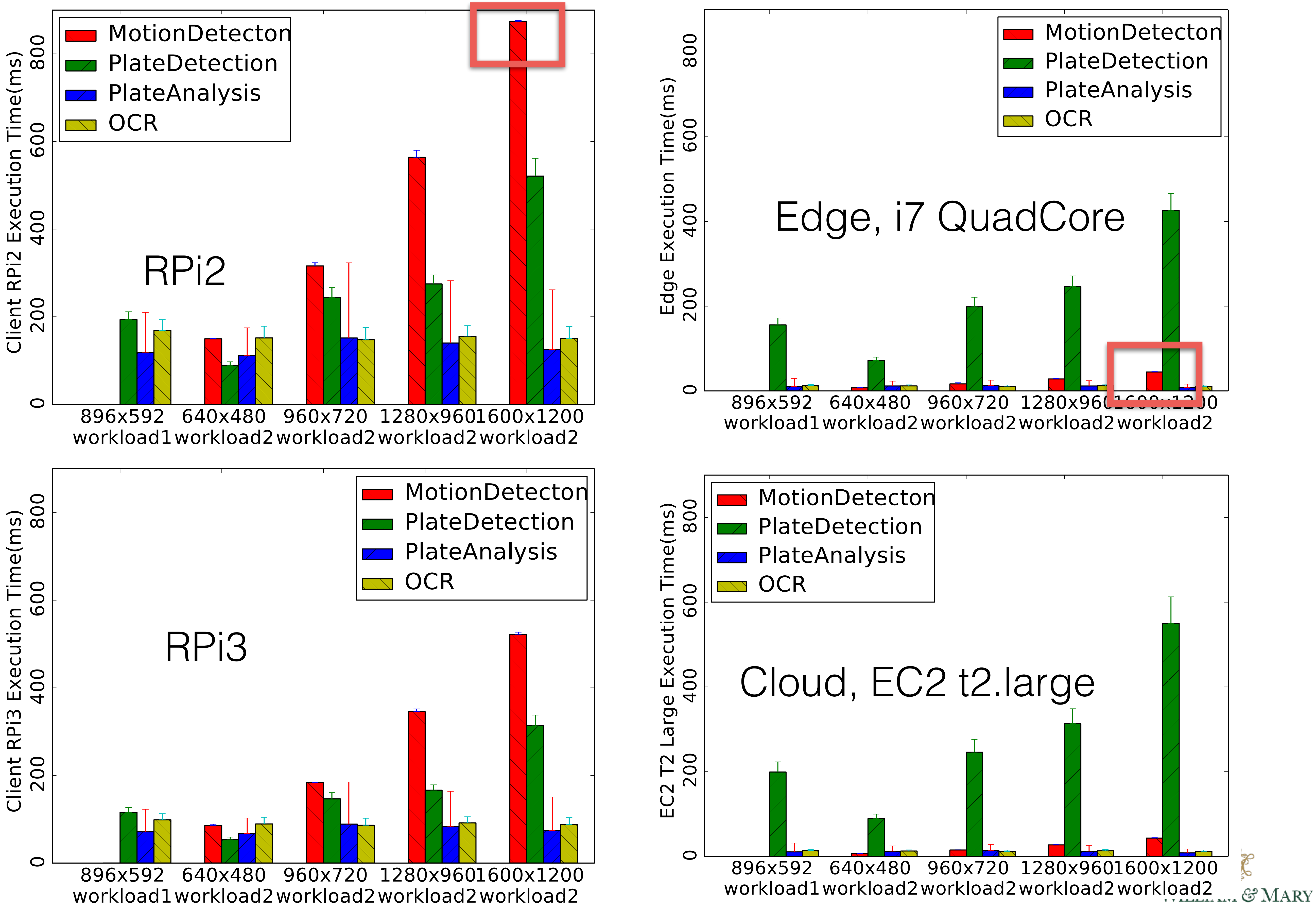
- Scheme
  - Naive schemes
    - Shortest Transmission Time First (STTF)
      - Periodical recalibrate the latency of transmitting data
        - Neglect existing workload of the neighbor
    - Shortest Queue Length First (SQLF)
      - Query nearby edge nodes about the task queue length
        - Neglect network latency
        - Scalability Issue: pull based
  - Our scheme
    - Shortest Scheduling Latency First (SSLF)
      - Dispatch special task to nearby edge nodes
        - When special task is executed, send response to edge-front node: push based
      - Predict the response time (regression analysis)
      - Piggyback update

- Introduction
- System Design Overview
- Edge Computing Services
- **Evaluation**
- Conclusion

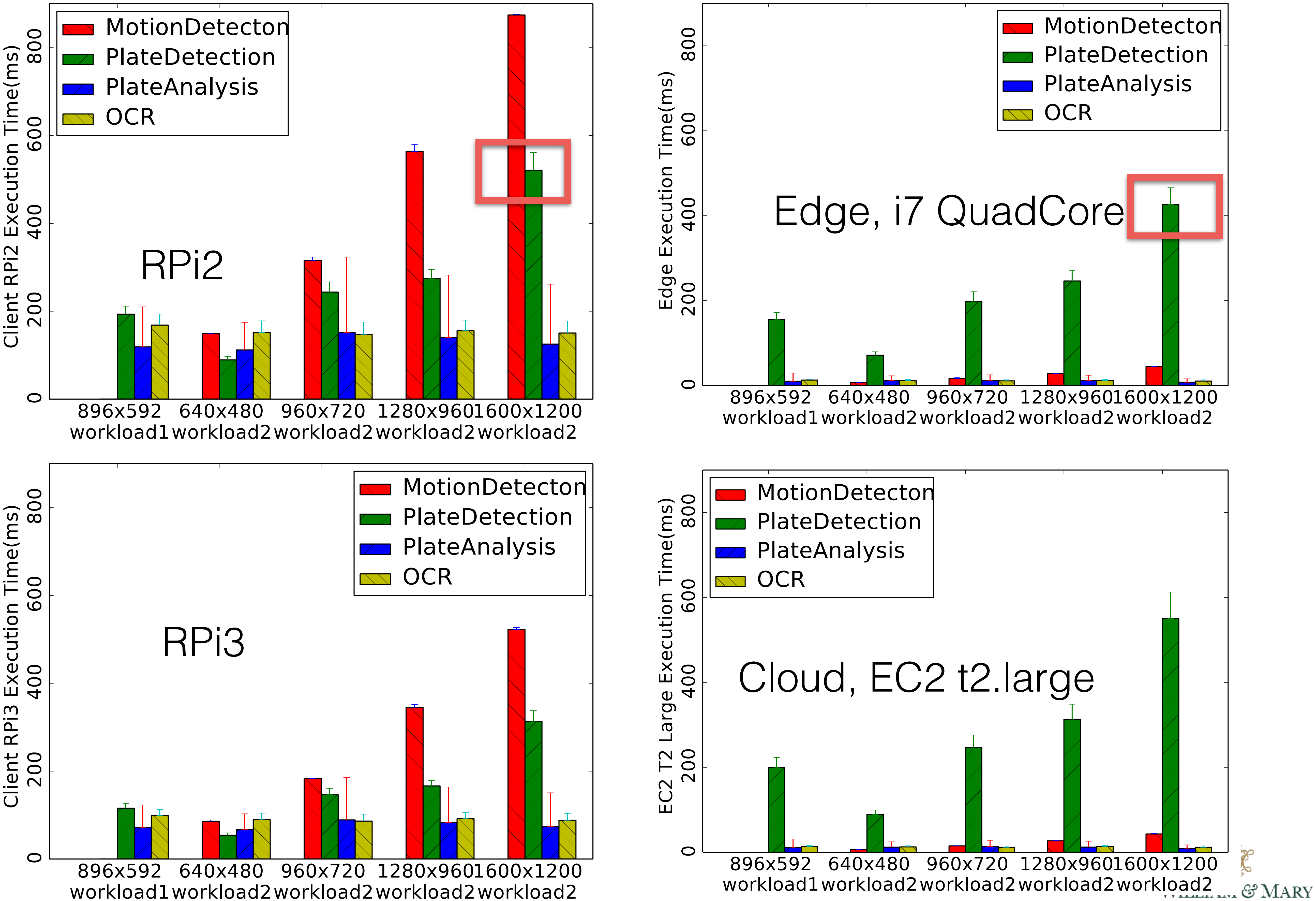
Evaluations - Setup

- Datasets
  - Caltech Vision Group 2001 testing image database - 126 images with resolution 896x592
  - Self-collected video containing license plates and converted into different resolutions (640x480, 960x720, 1280x960, 1600x1200)
  - Testing datasets in OpenALPR - 22 car images, with various resolutions (pixel 405x540 to 2514x1210, size 316 KB to 2.85 MB)
- Testbed
  - Four edge nodes, one as edge-front and three as neighbor edge nodes, cable connected, Quad-core CPU, 4GB Mem
  - Two types of client nodes: Raspberry Pi2 (cable) and Raspberry Pi 3 (Wifi)
  - Cloud node: t2.large EC2 instance

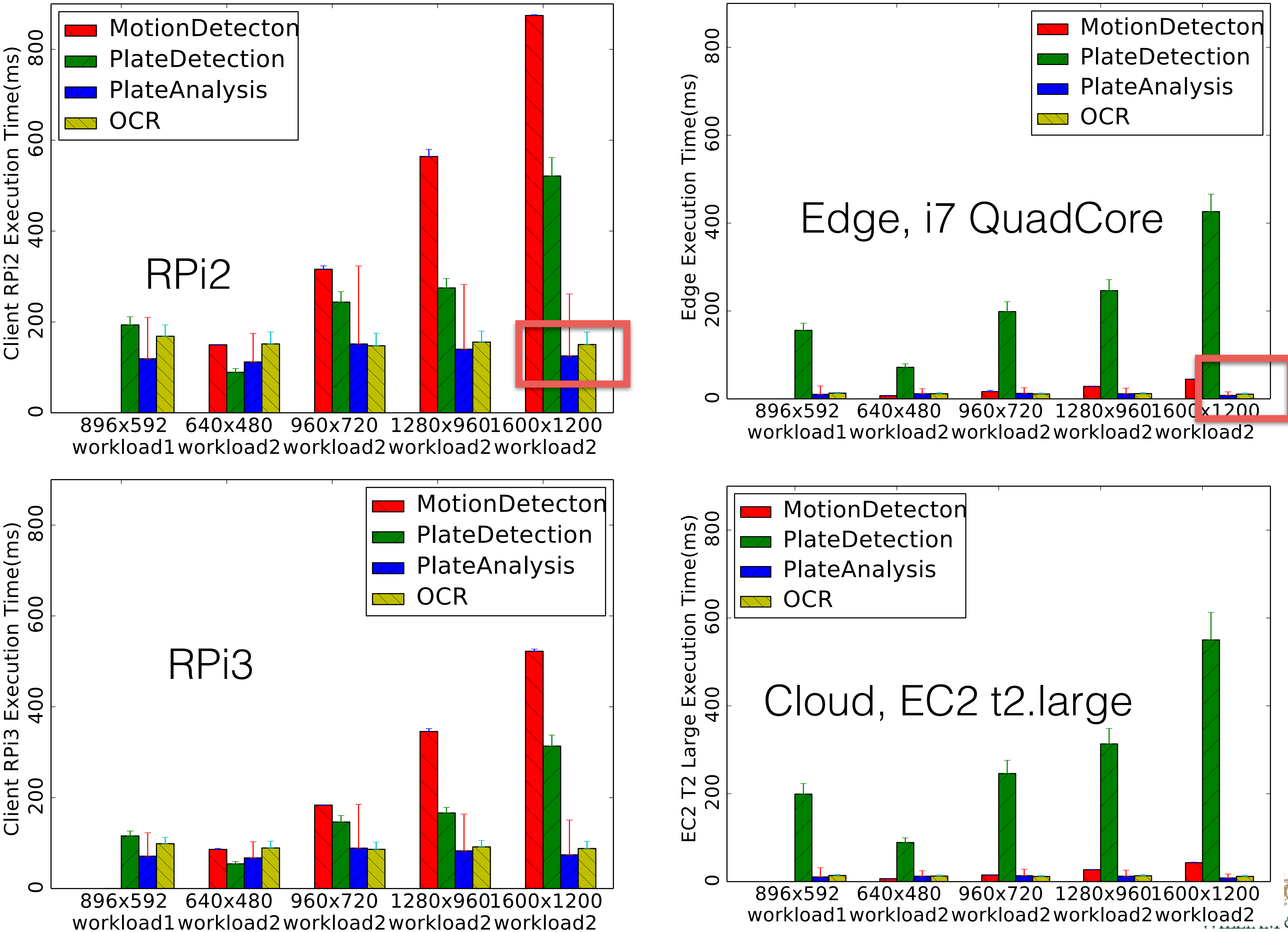
Evaluations - Task Profiling



Evaluations - Task Profiling

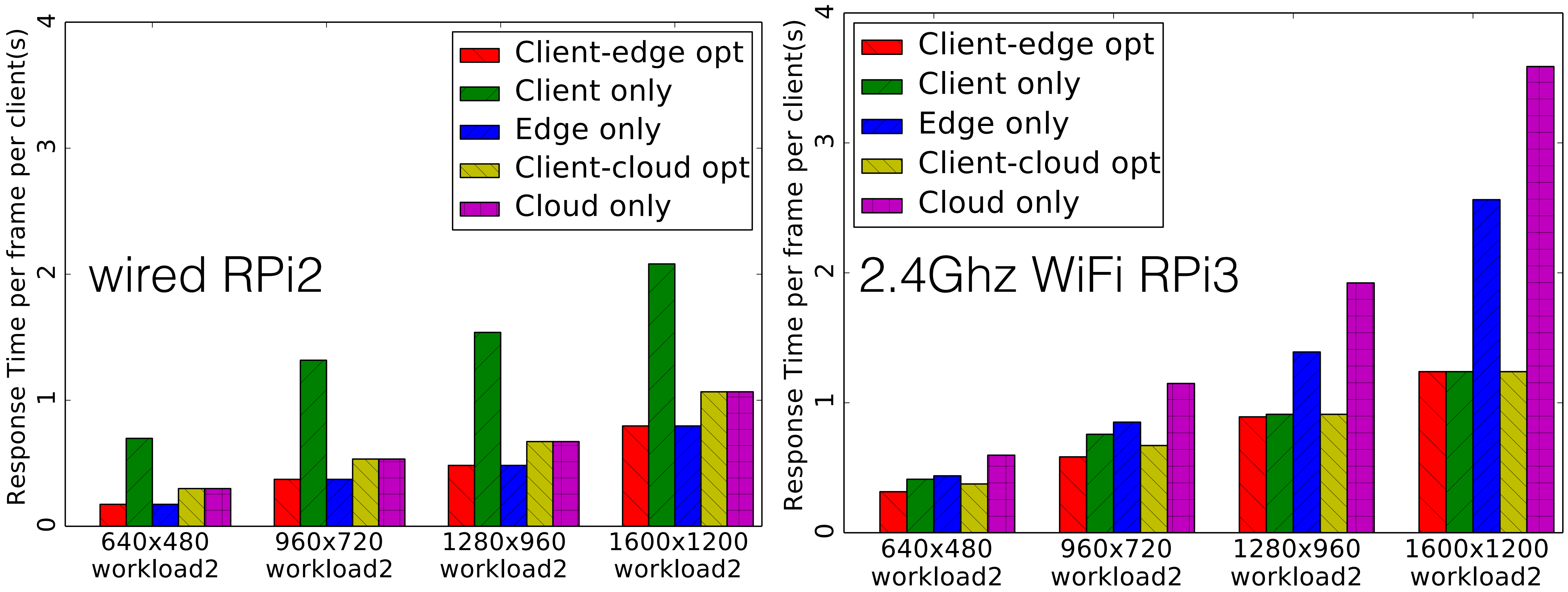


Evaluations - Task Profiling



W&M

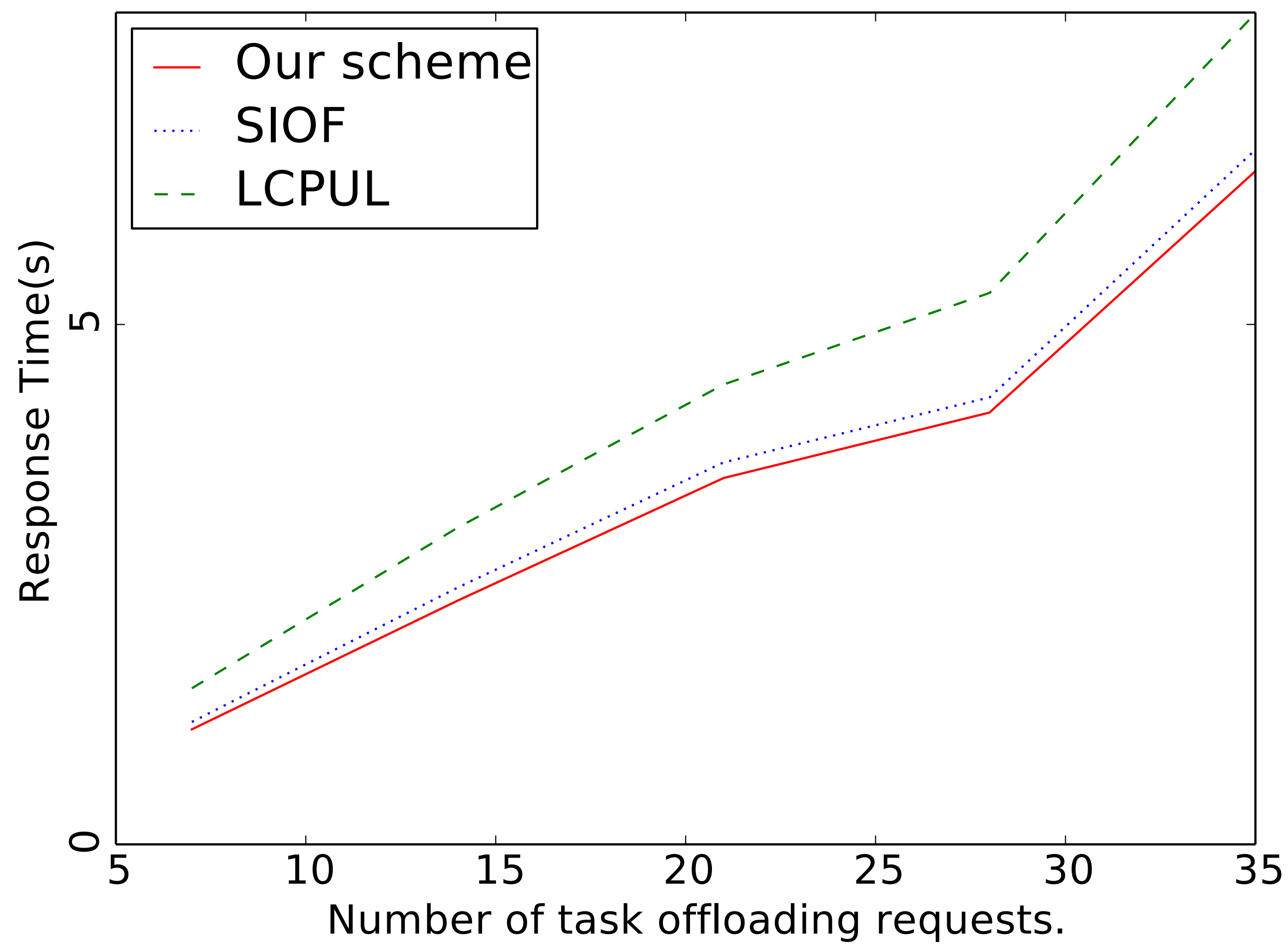
Evaluations - Task Offloading Selection



- Overall, by offloading tasks to an edge computing platform, the application we had chosen experienced a speedup up to 4.0x on wired client-edge configuration compared to local execution, and up to 1.7x compared to a similar client-cloud configuration.
- For clients with 2.4 GHz wireless interface, the speedup is up to 1.3x on client-edge configuration compared to local execution, and is up to 1.2x compared to similar client-cloud configuration .

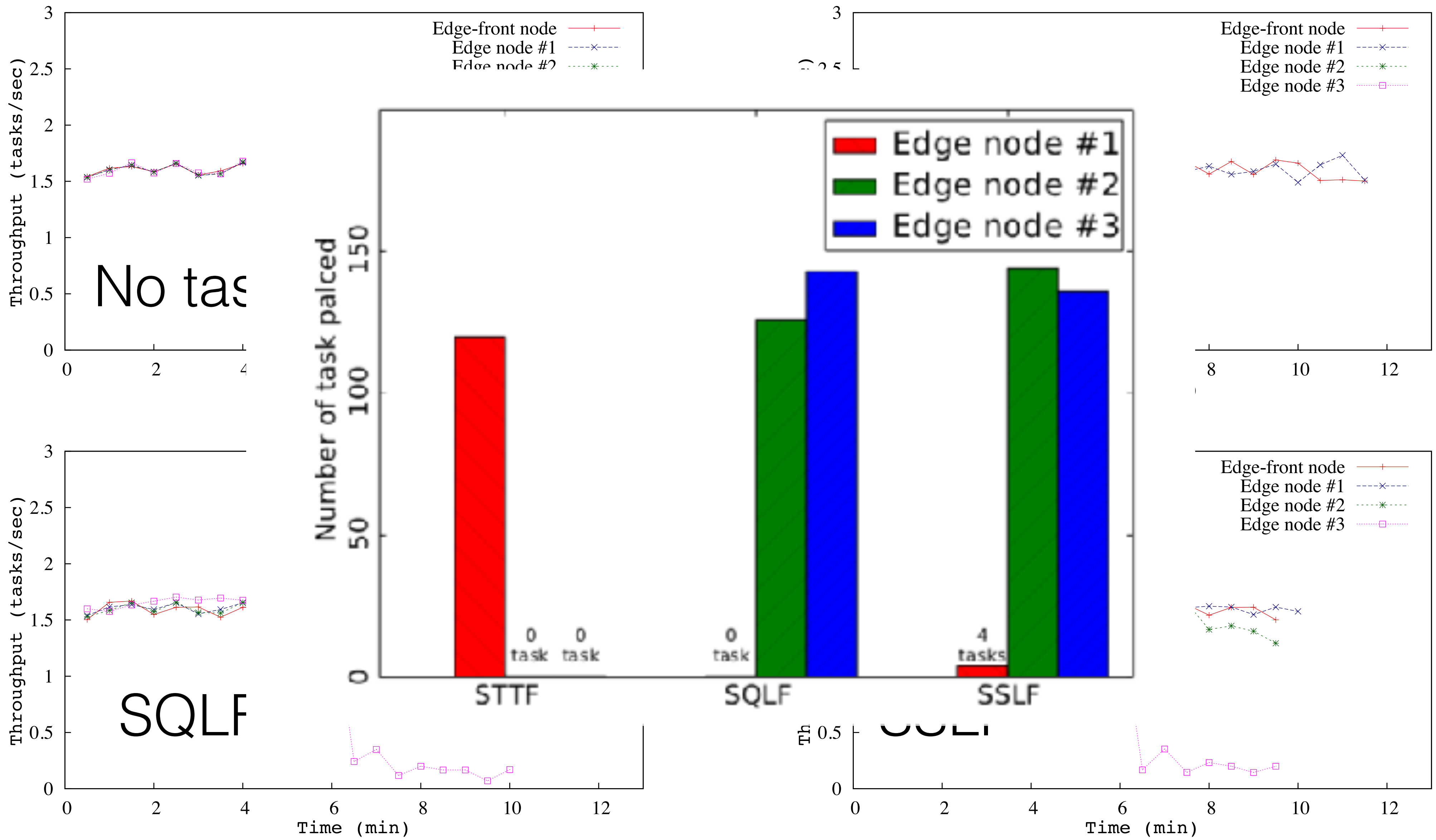
WILLIAM & MARY
CHARTERED 1693

Evaluations - Edge Task Queue Prioritizing



- Simulation
- Baselines: shortest IO first, longest CPU last
- Result shows LCPUL is the worst among three schemes and our scheme outperforms the shortest IO first scheme.

Evaluations — Inter-Edge Collaboration



- STTF scheme intend to place tasks to edge node with lowest transmission overhead but heaviest workload (node1)
- SQLF scheme intend to place tasks to edge node with lightest workload but with highest transmission overhead (node3)
- SSLF scheme considers both transmission time and the waiting time in the queue, therefore achieves the better performance.

WILLIAM & MARY
CHARTERED 1693

Conclusion

- We built LAVEA, a low-latency video edge analytic system
  - collaborates nearby client, edge and remote cloud nodes, and
  - transfers video feeds into semantic information at places closer to the users in early stages.
- We have formulated an optimization problem for offloading task selection and prioritized task queue to minimize the response time.
- In case of a saturating workload on the front edge node, we have proposed and compared various task placement schemes that are tailed for inter-edge collaboration.

End. Thank you.

Q&A