EdgeRM & Portkey: Resource Management and Adaptive Data Placement for Dynamic IoBTs

U.S. ARMY — DEVCOM ARMY RESEARCH LABORATORY — IoBT REIGN

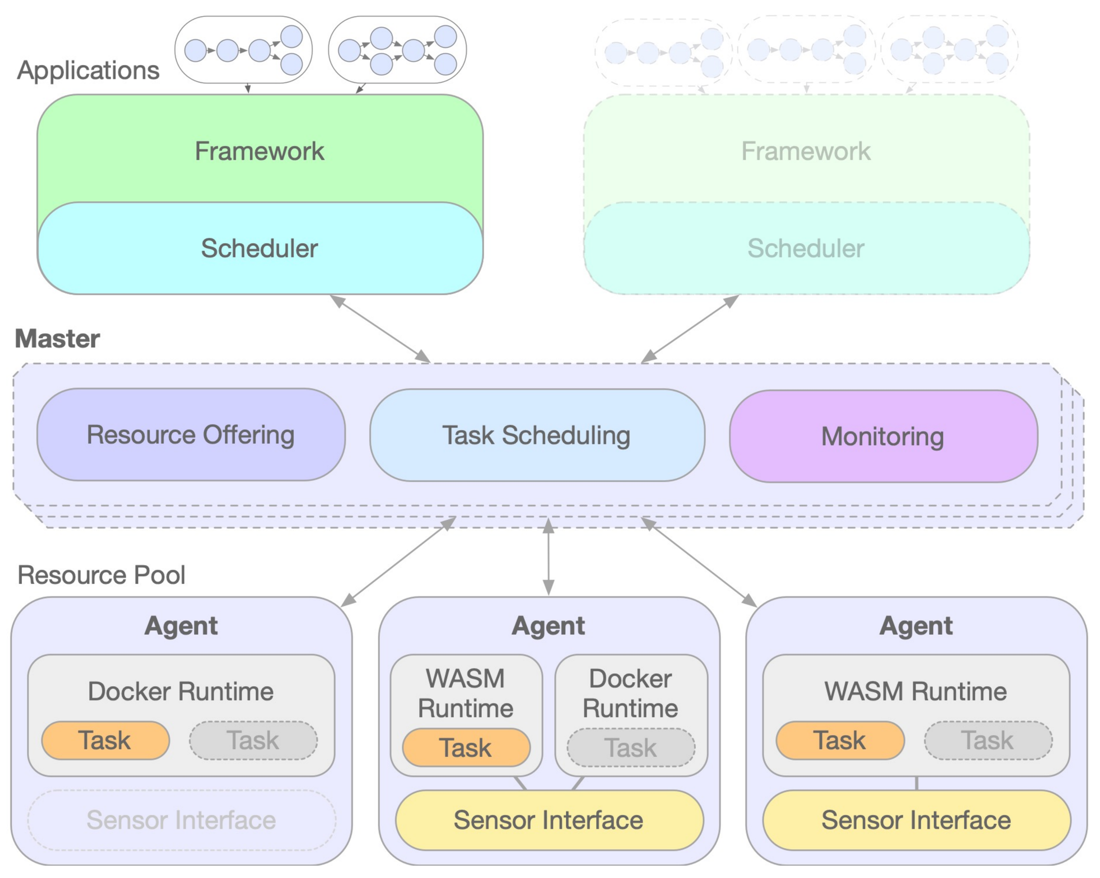
EdgeRM Objectives

- Extend distributed resource management frameworks to the heterogeneous IoBT domain
- Retain traditional resource management principles
- Integrate with novel resources (e.g. sensors, actuators)
- Support heterogeneous and potentially resource-constrained (i.e. non-Linux class) platforms
- Build a networked system that spans NATs.
- Improve maintainability and interactivity of networked clusters of IoBT assets.

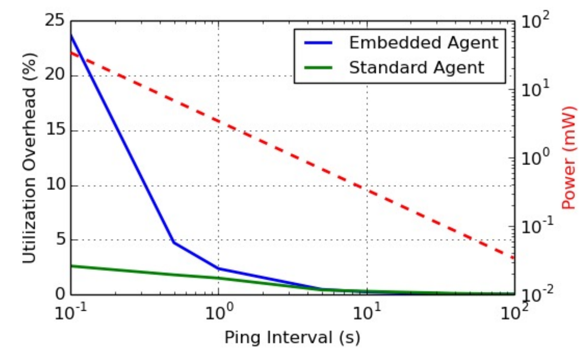
Approach

- Assets deploy an "agent" – a webserver client that connects to the EdgeRM Master to expose resources
- Embedded Agent for microcontroller-class devices
- Support Docker containers + WebAssembly modules
- Communication via protobufs over MQTT/CoAP/HTTP
- WebAssembly Sensor Interface for device access, configuration + control

EdgeRM System Architecture: Agent nodes expose available resources to the Master to be offered to multiple app frameworks



Minimal Agent Utilization + Power Overhead



Results

- Achieve unified cluster computing and sensor integration over devices with 128-256KB memory + 1MB of flash.
- WebAssembly provides interactivity with minimal latency overhead wrt native.
- Leverages growing compute capability wrt radio power consumption.
- We currently have a testbed at http://128.97.92.77:3000/ implementing two frameworks.

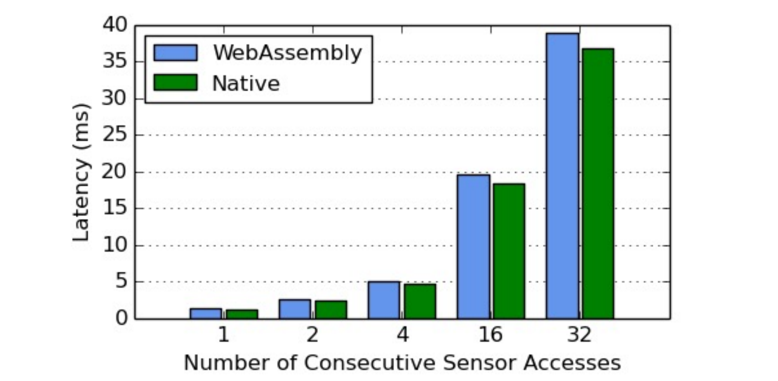
Path Forward

- Design placement and profiling integrations
- Expand classes of compute achievable on embedded assets (ML, waveform analysis)
- Integrate system stack from visual end-user interface to low-level task binding + execution

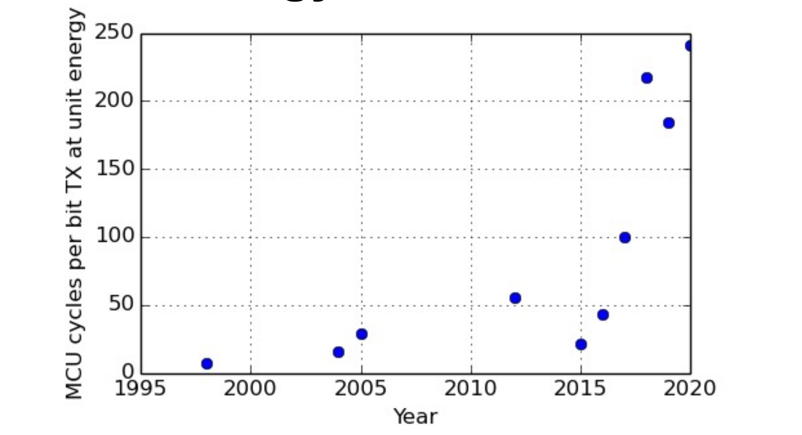
Embedded Agent Overhead

Code Segment	Text (B)	Data (B)	BSS (B)
Total	317,918	3,748	60,117
Agent Library	15,764	—	2,201
WAMR Runtime	51,564	—	1,250
Openthread (Net)	149116	—	33,119

WebAssembly Latency wrt Native



Increasing Compute vs Radio Energy Tradeoff



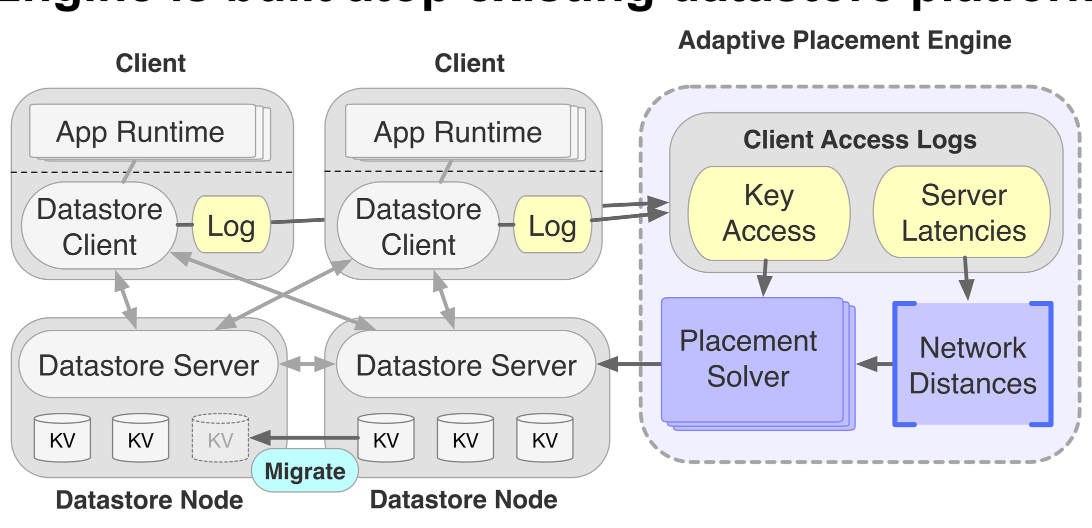
Portkey Objectives

- Develop a distributed data storage placement conducive to the rapid mobility of IoBT networks.
- Design a system that learns the appropriate data placement of key-value pairs based on access patterns.
- Provide open-source and readily-available implementation

Approach

- Formulate data placement as an online learning and optimization problem.
- Track client accesses to a distributed datastore cluster by injecting a usage profiler client-side leveraging lightweight sketching techniques.
- Make fast placement decisions over an intractable (NP-hard) partitioning problem that incorporates client accesses and available datastore host placements.
- Evaluate over a representative dataset of varying workloads exhibiting wide range of access patterns.

Portkey System Design: Adaptive Placement Engine is built atop existing datastore platform



Results

- Independent placement solver with greedy assignment operates in polynomial time enabling adaptive placement that responds quickly to IoBT dynamics.
- Latency improvement of up to 82%
- Lightweight sketching keeps overhead to a minimum (e.g. 8KB for 1024-node cluster)
- Implementation is readily available at https://github.com/nesl/AdaptiveKVPlacement

Placement Optimization Problem

$$C(k) = \min_{n} C_n(k) \quad \forall\, n \in N$$

$$C_n(k) = \sum_{i=1}^{N} f_i(k) \cdot d_{in}$$

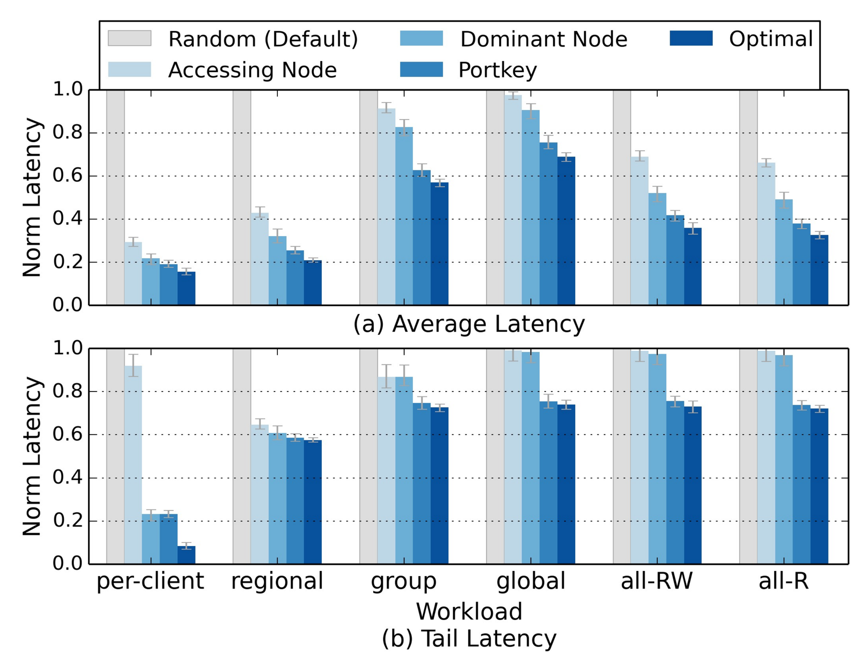
$$\overrightarrow{C(k)} = \overrightarrow{f(k)} \cdot \mathbf{D}$$

KV Independent Placement Solver Methodology

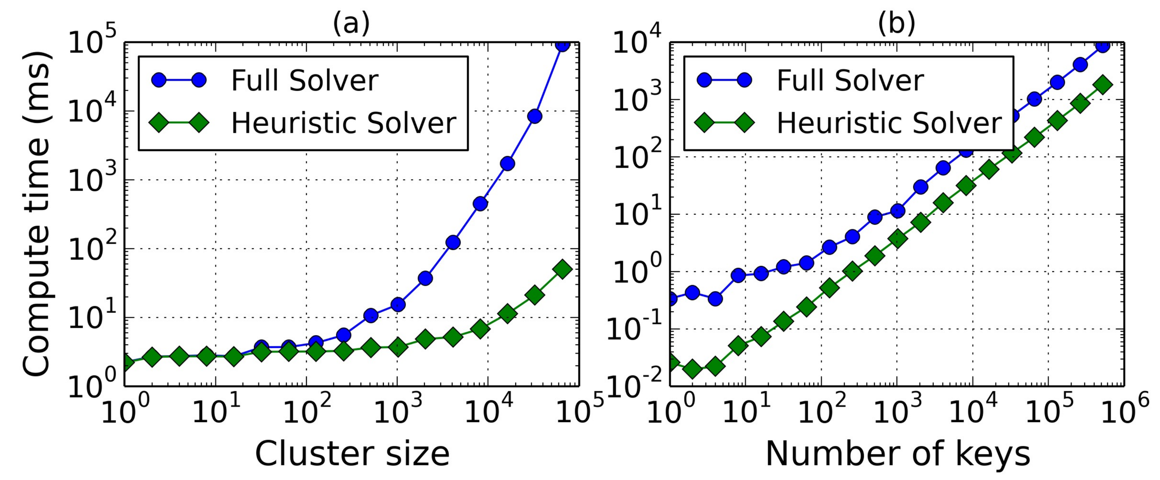
$$\begin{matrix} K_1 & [f_A \; f_B \; f_C \; f_D \; f_E] \\ K_2 & [f_A \; f_B \; f_C \; f_D \; f_E] \\ K_3 & [f_A \; f_B \; f_C \; f_D \; f_E] \\ K_4 & [f_A \; f_B \; f_C \; f_D \; f_E] \\ K_5 & [f_A \; f_B \; f_C \; f_D \; f_E] \end{matrix} \times \begin{bmatrix} d_{AA} & d_{AB} & d_{AC} & d_{AD} & d_{AE} \\ d_{BA} & d_{BB} & d_{BC} & d_{BD} & d_{BE} \\ d_{CA} & d_{CB} & d_{CC} & d_{CD} & d_{CE} \\ d_{DA} & d_{DB} & d_{DC} & d_{DD} & d_{DE} \\ d_{EA} & d_{EB} & d_{EC} & d_{ED} & d_{EE} \end{bmatrix} = \begin{matrix} [C_A \; C_B \; C_C \; C_D \; C_E] \\ [C_A \; C_B \; C_C \; C_D \; C_E] \\ [C_A \; C_B \; C_C \; C_D \; C_E] \\ [C_A \; C_B \; C_C \; C_D \; C_E] \\ [C_A \; C_B \; C_C \; C_D \; C_E] \end{matrix}$$

Key Access Patterns — Network Distance Matrix — Cost Matrix

Portkey reduces average latency by 21-82% and tail latency by 26-77% over existing strategies



Solver Scalability: $\mathcal{O}(kn^2)$



Intuition: Given high mobility, prioritize fast placement over optimal, iterate to adapt

ILLINOIS — USC University of Southern California — UCLA — UMass Amherst — Carnegie Mellon University — Georgetown University — SRI International — Stanford University — The University of Texas at Austin — DEVCOM Army Research Laboratory

POINT OF CONTACT:
Joseph Noor
jnoor@cs.ucla.edu
714-393-3196