

# A Document-Oriented Robot Memory for Knowledge Sharing and Hybrid Reasoning on Mobile Robots

**Frederik Zwilling**

Supervisors: Prof. Lakemeyer, PhD., Prof. Dr. Jarke  
Adviser: Tim Niemueller



## **Database to store information of a robot about its environment**

- **Domain:** Logistics and domestic service robotics
- **Purpose:** Scalable storage and rich querying
- **Focus:** Knowledge sharing and hybrid reasoning for knowledge-based systems

## 1 Motivation

## 2 Background

## 3 Related Work

## 4 Approach

## 5 Implementation

## 6 Evaluation

## 7 Conclusion

# Why do robots need a memory?



- Store and reason about world state
- Share information in multi-robot system
- Remember object sights
- Persistent storage
- Consistent information base for different components

# Robot Memory Goals

- Flexible storage and retrieval
- Spatio-temporal grounding
- Persistent storage
- Memory sharing between knowledge-based systems
- Distributed memory for multi-robot systems
- Computation on demand (*Computables*)
- Notification about updates (*Triggers*)

**1** Motivation

**2** Background

**3** Related Work

**4** Approach

**5** Implementation

**6** Evaluation

**7** Conclusion

# Application Domains



## RoboCup Logistics League

- Production logistics  
in smart factory
- Share world model
  - between robots
  - between global planner  
and reasoner executive

# Application Domains



## RoboCup Logistics League

- Production logistics in smart factory
- Share world model
  - between robots
  - between global planner and reasoner executive



## RoboCup@Home

- Domestic service robots
- Collect information about concrete environment
- Hybrid reasoning with spatio-temporal knowledge

# Planners and Reasoners in Fawkes

## CLIPS Rules Engine

- First-Order Logic forward chaining system
- Fact base and condition-action rules
- ⇒ Used for world model reasoning and execution monitoring



# Planners and Reasoners in Fawkes

## CLIPS Rules Engine

- First-Order Logic forward chaining system
- Fact base and condition-action rules
- ⇒ Used for world model reasoning and execution monitoring



## Planning Domain Definition Language (PDDL)

- Standardized language for planning problems
- Find action sequence through heuristic search
- ⇒ Used for finding global plans

# Planners and Reasoners in Fawkes

## CLIPS Rules Engine

- First-Order Logic forward chaining system
- Fact base and condition-action rules
- ⇒ Used for world model reasoning and execution monitoring



## Planning Domain Definition Language (PDDL)

- Standardized language for planning problems
- Find action sequence through heuristic search
- ⇒ Used for finding global plans

## Motion Planners

- Robot arm and locomotion collision avoidance
- Depend on geometric data



**1** Motivation

**2** Background

**3** Related Work

**4** Approach

**5** Implementation

**6** Evaluation

**7** Conclusion

# Robot Information Storage Systems

## KnowRob [Tenorth and Beetz, 2013]

- Common sense reasoning with ontologies
- Based on Prolog
- Virtual knowledge base to interface perception

## OpenRobots Ontology (ORO) [Lemaignan et al., 2010]

- Common sense reasoning with ontologies
- Based on Java
- Events notifying about changes

# Robot Information Storage Systems

## KnowRob [Tenorth and Beetz, 2013]

- Common sense reasoning with ontologies
- Based on Prolog
- Virtual knowledge base to interface perception

## OpenRobots Ontology (ORO) [Lemaignan et al., 2010]

- Common sense reasoning with ontologies
- Based on Java
- Events notifying about changes

- Not applicable for multi-robot systems
- Scalability, efficiency concerns
- Missing events / computation on demand

# Document Orientation

- Documents: Sets of key-value pairs
- Java Script Object Notation (JSON)

```
{  
    "key": "value",  
    "subdocument": { "x":3, "y":1},  
    "array": [{"n":0.1}, {"n":2}]  
}
```

- Denormalized (information bundled in documents)
  - Schema free
- ⇒ Allows generic, flexible, and distributable robot memory

# MongoDB Database System



- Scalable and widely used
- Query language with aggregation, MapReduce, JavaScript
- Indexing for fast querying
- Distributable with Replica Sets  
Operations Log (Oplog) to forward changes
- Comparable good performance and scalability  
[Oliveira and del Val Cura, 2016, Li and Manoharan, 2013]

# Related Work with MongoDB

## Robot Database with MongoDB [Niemueller et al., 2012]

- Data logging for evaluation and fault analysis
- Generic and scalable storage with MongoDB
- Integration in Fawkes and ROS

## Extensions of MongoDB

- Triggers with replication Oplog [Dwivedi and Dubey, 2016]

**1** Motivation

**2** Background

**3** Related Work

**4** Approach

**5** Implementation

**6** Evaluation

**7** Conclusion

# Computables and KBS Interface



Perception

( at cup kitchen\_counter )

PDDL-based Planner

# Computables and KBS Interface



{ object: "cup", position: {x: 8, y:4} }

( at cup kitchen\_counter )



# Computables and KBS Interface



{ object: "cup", position: {x: 8, y:4} }

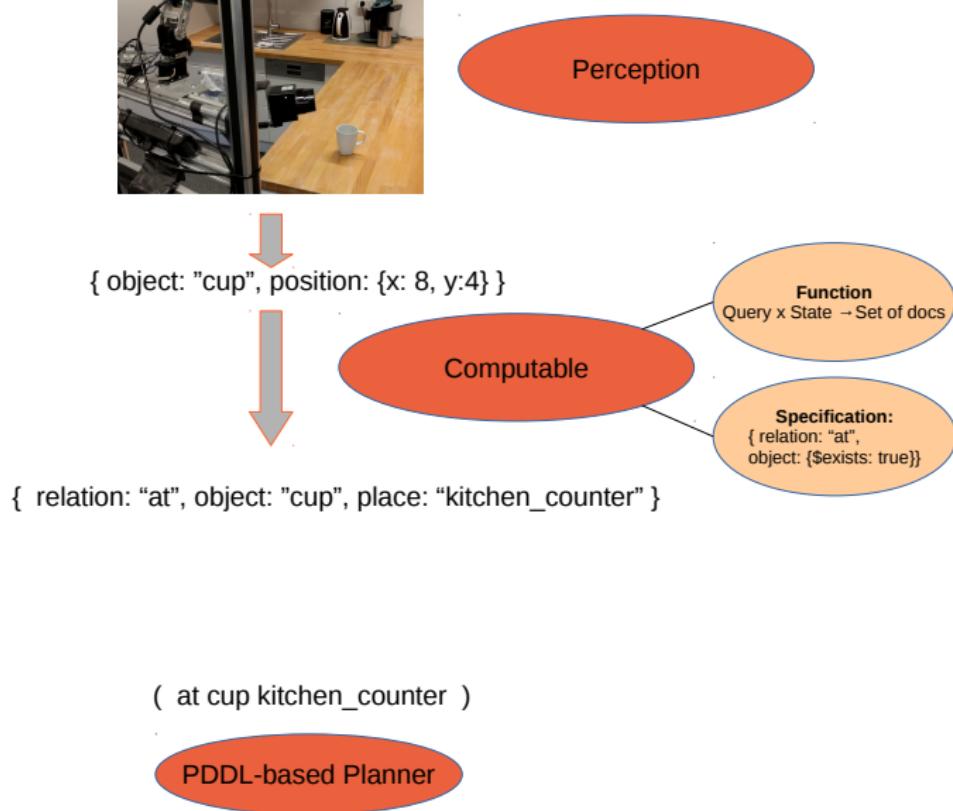


{ relation: "at", object: "cup", place: "kitchen\_counter" }

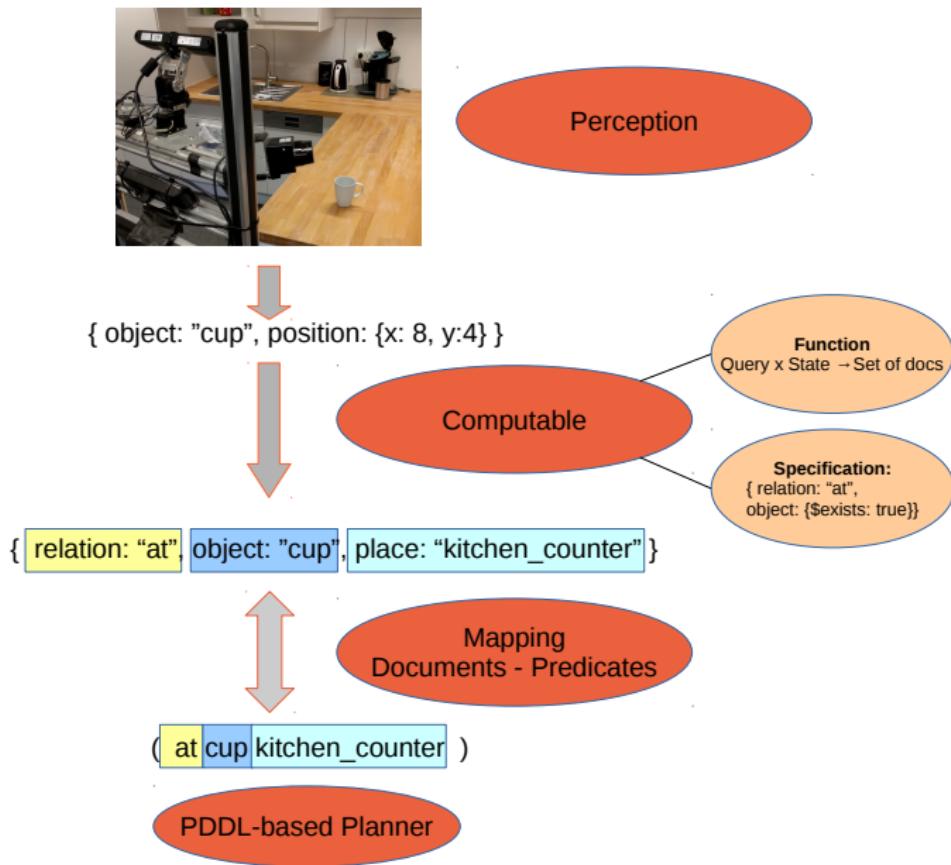
( at cup kitchen\_counter )

PDDL-based Planner

# Computables and KBS Interface



# Computables and KBS Interface



# Triggers



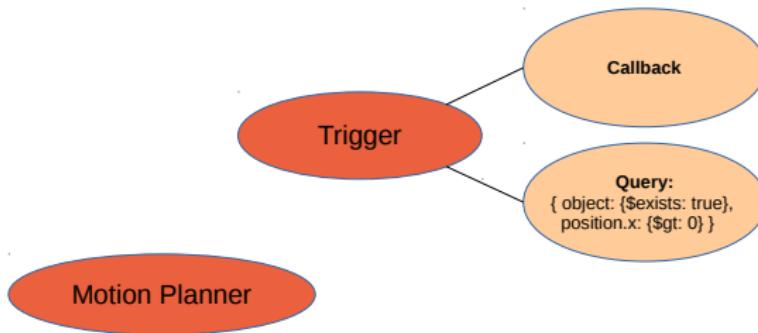
{ object: "cup", position: {x: 8, y:4} }



# Triggers



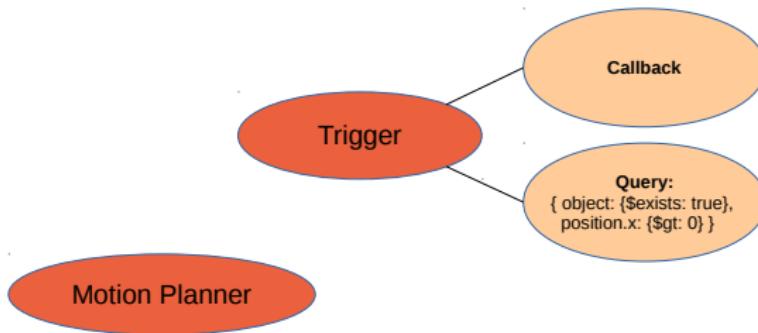
{ object: "cup", position: {x: 8, y:4} }



# Triggers



```
{ object: "cup", position: {x: 8, y:4} }  
{ object: "cereal", position: {x: 8, y:3} }
```



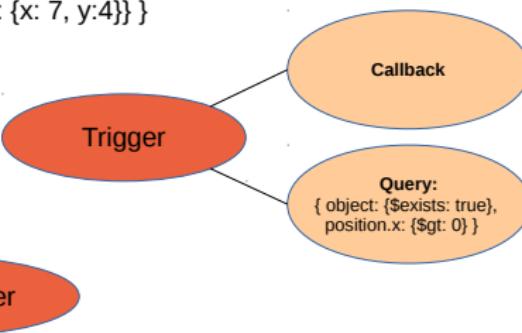
# Triggers



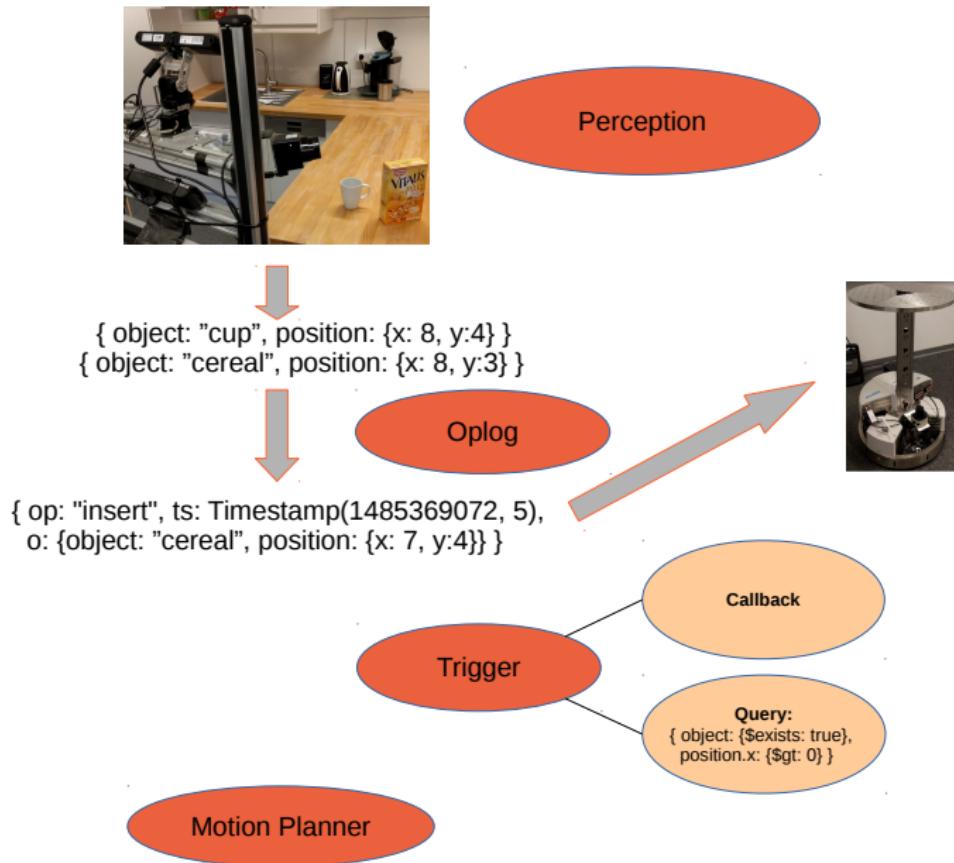
{ object: "cup", position: {x: 8, y:4} }  
{ object: "cereal", position: {x: 8, y:3} }



{ op: "insert", ts: Timestamp(1485369072, 5),  
o: {object: "cereal", position: {x: 7, y:4}} }



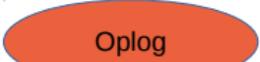
# Triggers



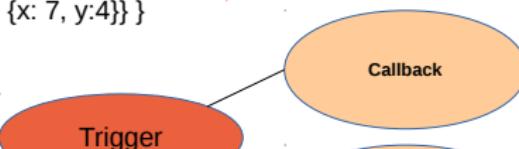
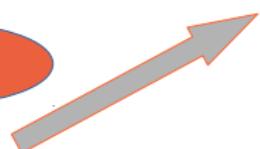
# Triggers



{ object: "cup", position: {x: 8, y:4} }  
{ object: "cereal", position: {x: 8, y:3} }

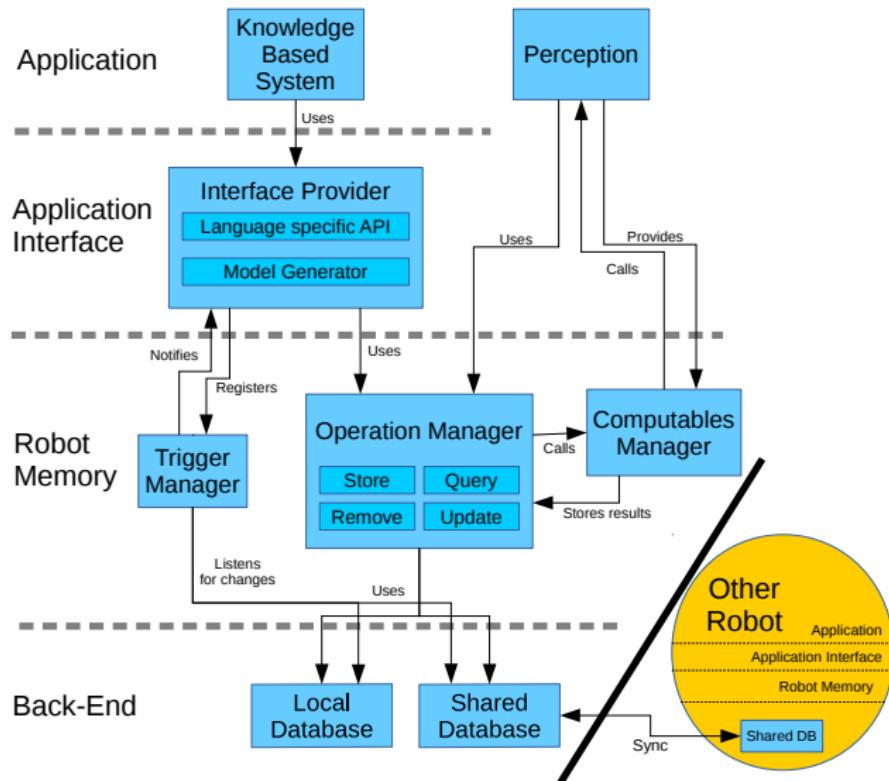


{ op: "insert", ts: Timestamp(1485369072, 5),  
o: {object: "cereal", position: {x: 7, y:4}} }

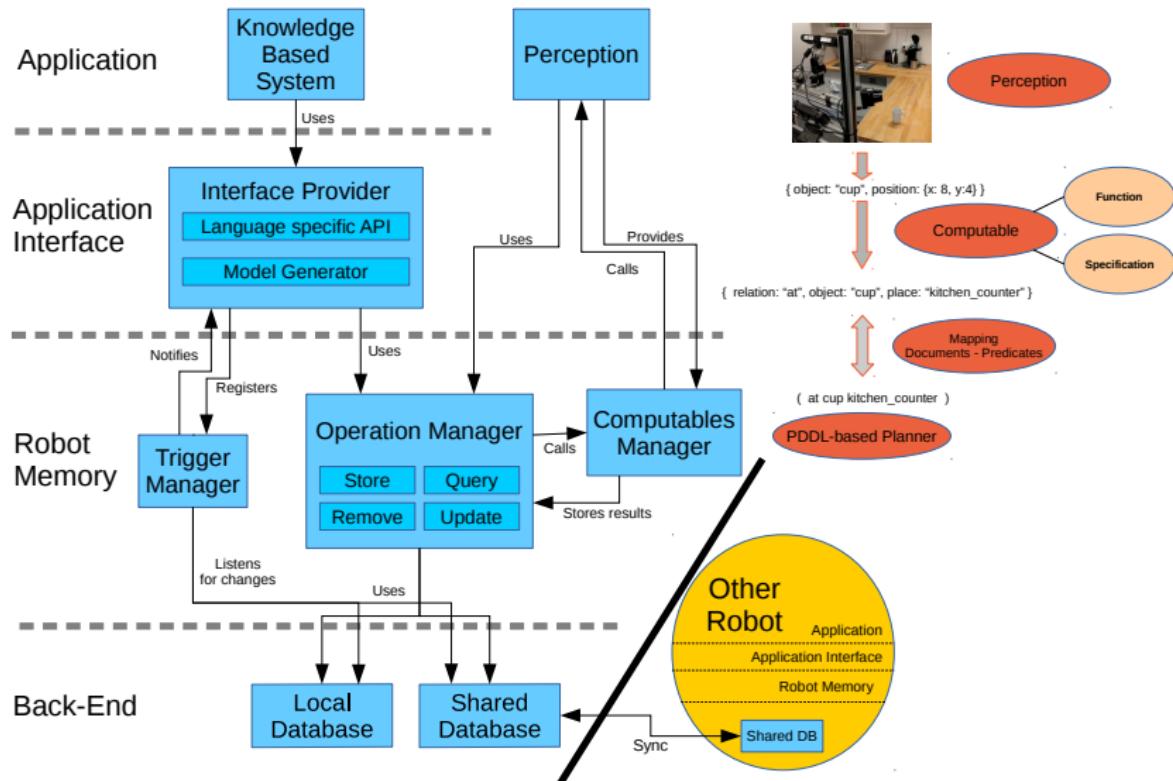


Query:  
{ object: {\$exists: true},  
position.x: {\$gt: 0} }

# Architecture



# Architecture



**1** Motivation

**2** Background

**3** Related Work

**4** Approach

**5** Implementation

**6** Evaluation

**7** Conclusion

# Back-End and Robot Memory

## Distribution in multi-robot system

- Write operations only on primary instance
- Eventual consistency on secondaries

## Robot Memory

- Caching of computed documents
- Automated start-up of MongoDB

# CLIPS Interface

## CLIPS Characteristics

- Fact base as working memory

```
(cap-station (name M-CS1) (loaded NONE) (caps-on-shelf 3))
```

- Condition-action rules
- Procedural functions

# CLIPS Interface

## CLIPS Characteristics

- Fact base as working memory

```
(cap-station (name M-CS1) (loaded NONE) (caps-on-shelf 3))
```

- Condition-action rules
- Procedural functions

## Robot Memory Interface

- Provide operation and traversal functions in CLIPS
- Mapping between facts and documents

```
{ relation: "cap-station", name: "M-CS1",
  loaded: "NONE", caps-on-shelf: NumberLong(3) }
```

- Assert trigger events as facts

# PDDL Interface

## PDDL Characteristics

- Domain definition and problem description as input
- Predicates represent information

```
(:goal (on A B))  
(:init (on-table A) (on-table B))
```

# PDDL Interface

## PDDL Characteristics

- Domain definition and problem description as input
- Predicates represent information

```
(:goal (on A B))  
(:init (on-table A) (on-table B))
```

## Robot Memory Interface

- Mapping of documents to predicates
- Generation of problem description from template

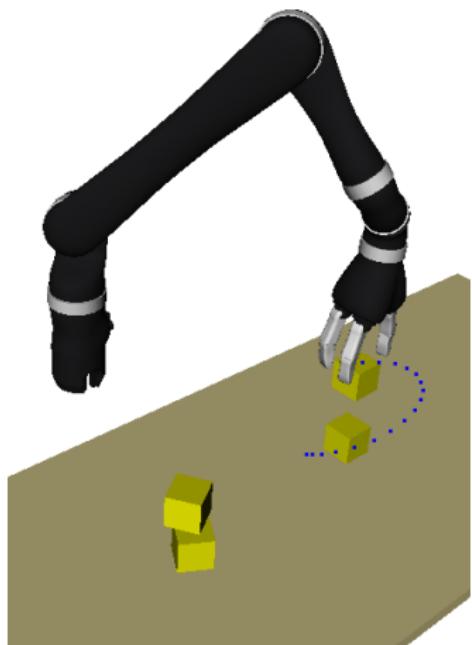
```
(:goal <><>GOAL>>  
(:init <><>#ONTABLE|{relation:'on-table'}>>  
      (on-table <><>object>>) <><>/ONTABLE>>))
```

```
{ relation: "on-table", object: "A" },  
{ relation: "on-table", object: "B" }
```

# OpenRAVE Interface

## OpenRAVE Characteristics

- Requires geometric information
- Operates in motion planner scene



# OpenRAVE Interface

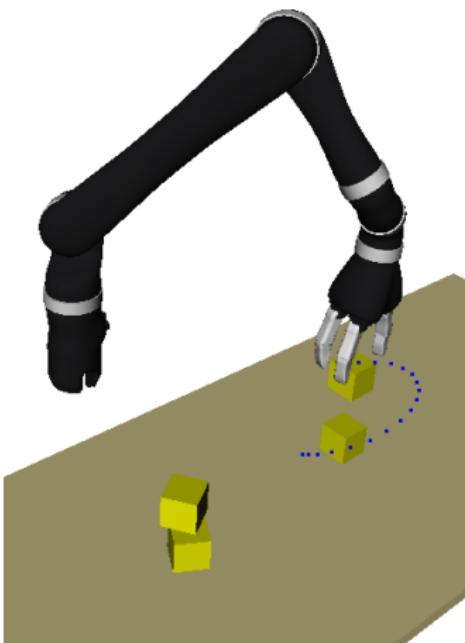
## OpenRAVE Characteristics

- Requires geometric information
- Operates in motion planner scene

## Robot Memory Interface

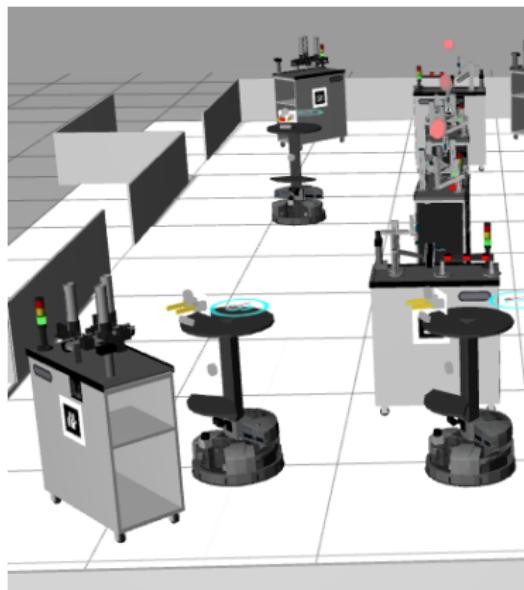
- Updates planner scene  
based on positions  
represented in documents

```
{  
    block: "B",  
    frame: "map",  
    translation:  
        [0.43, -0.04, 0.01]  
}
```

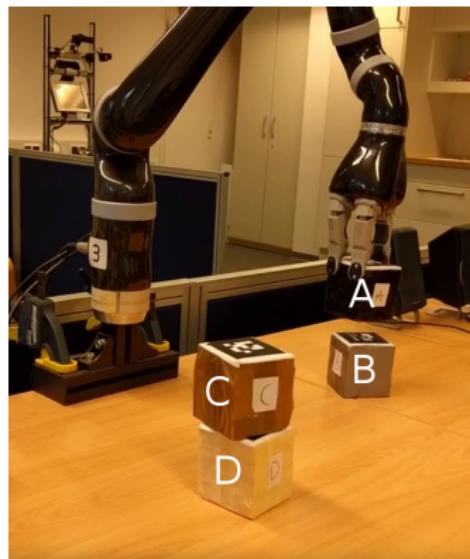


# Application and evaluation scenarios

## World model synchronization between robots in the RCLL



## Blocks world with a robot arm



**1** Motivation

**2** Background

**3** Related Work

**4** Approach

**5** Implementation

**6** Evaluation

**7** Conclusion

# Qualitative Evaluation

## Experience from evaluation scenarios

- Flexible storage and powerful querying
- Convenient memory sharing between KBS and in distributed system
- Allows hybrid reasoning with computables (e.g. on-table derived from geometric position)
- Triggers useful for world model updates and messages

# Qualitative Evaluation

## Experience from evaluation scenarios

- Flexible storage and powerful querying
  - Convenient memory sharing between KBS and in distributed system
  - Allows hybrid reasoning with computables (e.g. on-table derived from geometric position)
  - Triggers useful for world model updates and messages
- 
- Beneficial for AI/robot software development
  - Especially for combining different planners/reasoners

# Qualitative Evaluation: Limitations

## Trade-offs / Limitations

- Trigger only for changes of single documents
- No direct trigger evaluation for computables
- Query complexity determined by application

# Quantitative Evaluation

## Tidy up scenario

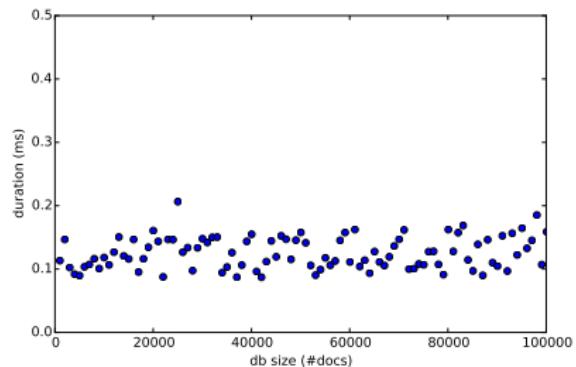
- Robot Memory with information about objects

```
{ name: "coffee machine",
  position: "counter",
  tidied: "counter" },
{ name: "milk",
  position: "counter",
  tidied: "fridge" }
```

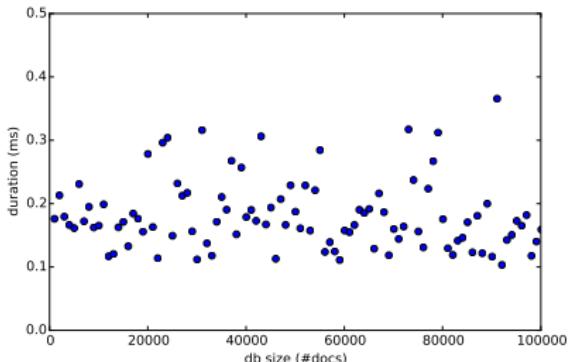
- Measure operation durations with increasing domain size
- With / without indexing



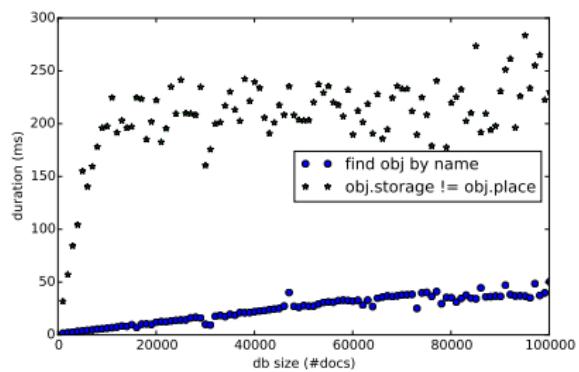
# Duration of Robot Memory Operations I



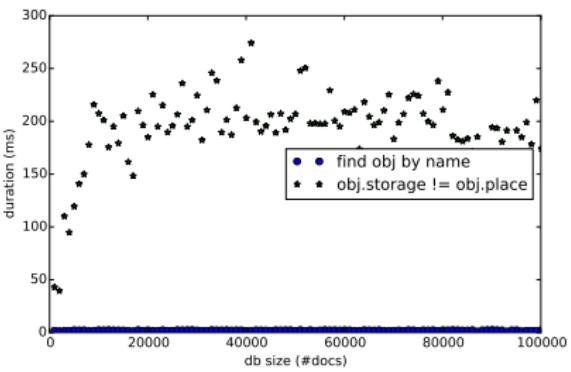
Insertions



Insertions with Indexing

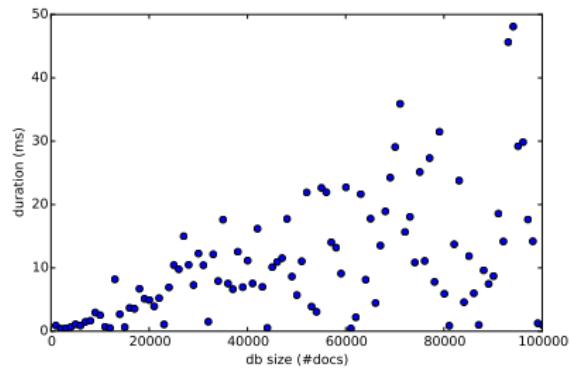


Queries

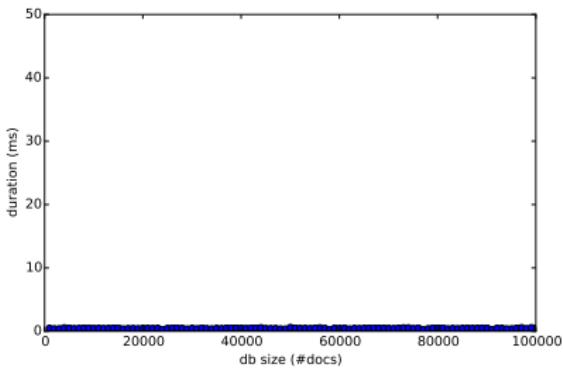


Queries with Indexing

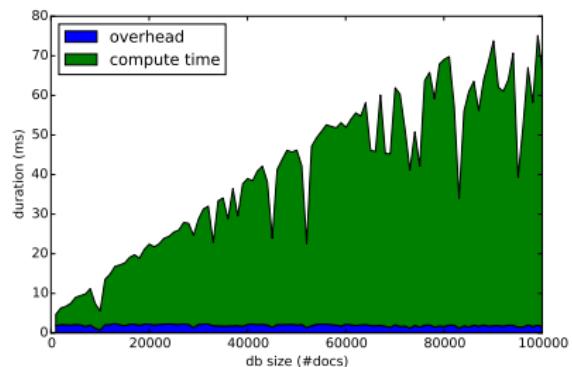
# Duration of Robot Memory Operations II



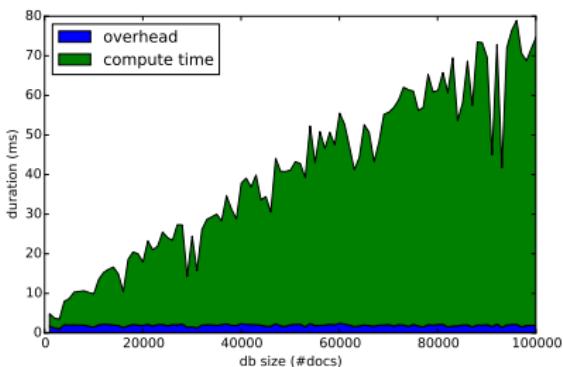
Updates



Updates with Indexes



Computables



Computables with Indexes

**1** Motivation

**2** Background

**3** Related Work

**4** Approach

**5** Implementation

**6** Evaluation

**7** Conclusion

# Foundation for Future Projects

## Two projects currently using the Robot Memory

- Both for centralized global task planning in the RCLL
- Reactive ASP and real-time constraints
- PDDL with temporal aspects

# Foundation for Future Projects

## Two projects currently using the Robot Memory

- Both for centralized global task planning in the RCLL
- Reactive ASP and real-time constraints
- PDDL with temporal aspects

## Beneficial features

- Distributed memory shared by planner and executives
- CLIPS agent integration
- Triggers for notifications
- PDDL problem definition generation

# Conclusion and Questions

## Generic Robot Memory

**flexible storage and expressive querying for hybrid reasoning and world model sharing between different KBS**

# Conclusion and Questions

## Generic Robot Memory

**flexible storage and expressive querying for hybrid reasoning and world model sharing between different KBS**

- Document-oriented representation and querying
- Distributable and persistent
- Interfaces for KBS
- Triggers for notification
- Computables
- Symbolic/spatio-temporal
- Beneficial and efficient in application scenarios
- Foundation for future projects

# References I

-  Dwivedi, K. and Dubey, S. K. (2016).  
Implementation of Data Analytics for MongoDB Using Trigger Utility.  
In *Computational Intelligence in Data Mining—Volume 1*, pages 39–47. Springer.
-  Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., and Beetz, M. (2010).  
ORO, a knowledge management platform for cognitive architectures in robotics.  
In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3548–3553. IEEE.

# References II

-  Li, Y. and Manoharan, S. (2013).  
A performance comparison of SQL and NoSQL databases.  
In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 15–19. IEEE.
-  Niemueller, T., Lakemeyer, G., and Srinivasa, S. S. (2012).  
A Generic Robot Database and its Application in Fault Analysis and Performance Evaluation.  
In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.
-  Oliveira, F. R. and del Val Cura, L. (2016).  
Performance Evaluation of NoSQL Multi-Model Data Stores in Polyglot Persistence Applications.  
In *Proceedings of the 20th International Database Engineering & Applications Symposium*, pages 230–235. ACM.

# References III

- 
- Tenorth, M. and Beetz, M. (2013).
- 
- KnowRob – A Knowledge Processing Infrastructure for
- 
- Cognition-enabled Robots. Part 1: The KnowRob System.
- 
- Int. Journal of Robotics Research (IJRR)*
- .

# Theoretical Foundation

---

## Definition of documents representing knowledge

e.g.  $\{("object", "cup"), ("room", "kitchen"), ("position", \{("x", 8), ("y", 4)\})\}$

1. **Keys:**  $\mathcal{K} := \Sigma^*$
2. **Atomic values:**  $\mathcal{V}_0$  are constants
3. **Unnested key-value pairs:**  $\mathcal{P}_0 := \mathcal{K} \times \mathcal{V}_0$
4. **Unnested documents:**

$$\mathcal{D}_0 := \{d \in \mathbb{P}(\mathcal{P}_0) | \forall (k, v), (k', v') \in d, k \neq k' \vee (k, v) = (k', v')\}$$

## With nesting

1. **Values:**  $\mathcal{V}_n := \mathcal{V}_{n-1} \cup \mathcal{D}_{n-1}$
2. **Key-Value Pairs:**  $\mathcal{P}_n := \mathcal{K} \times \mathcal{V}_n$
3. **Documents:**

$$\mathcal{D}_n := \{d \in \mathbb{P}(\mathcal{P}_n) | \forall (k, v), (k', v') \in d, k \neq k' \vee (k, v) = (k', v')\}$$

**Finitely nested documents:**  $\mathcal{D} = \bigcup_{n \in \mathbb{N}} \mathcal{D}_n$

**Values:**  $\mathcal{V} = \bigcup_{n \in \mathbb{N}} \mathcal{V}_n$

# Theoretical Foundation

## Definition Robot Memory

1. **Database:** finite set  $\mathcal{DB} \subset \mathcal{D}$
2. **Query:** represented by document  $q \in \mathcal{D}$   
yields set of documents  $r \subset \mathcal{DB}$  as result  
e.g.  $q = \{("object", "cup"), ("room", "kitchen")\}$
3. **Computable:**  $f : \mathcal{D} \rightarrow \mathbb{P}(\mathcal{D})$
4. **Set of Computables:**  $\mathcal{C}$
5. **Robot Memory:**  $\mathcal{RM} = (\mathcal{DB}, \mathcal{C})$
6. **Memorized Documents:**  $mem(\mathcal{RM}) = \mathcal{DB} \cup \bigcup_{f \in \mathcal{C}} f(\mathcal{D})$

# Theoretical Foundation

## Mapping into PDDL

e.g.  $map_p(\{("predicate", "at"), ("object", "cup"), ("room", "kitchen")\})$   
 $= at(map_f("cup"), map_f("kitchen")) = at(cup, kitchen).$

1. **Predicate symbols  $\mathcal{R}$ , Function symbols  $\mathcal{F}$**
2. **Name mapping:**

$$name_{pred} : \mathcal{R} \rightarrow \Sigma^* \text{ and } name_{func} : \mathcal{F} \rightarrow \Sigma^*$$

$$name_{pred-atr} : \mathcal{R} \times \mathbb{N} \rightarrow \mathcal{K} \text{ and } name_{func-atr} : \mathcal{F} \times \mathbb{N} \rightarrow \mathcal{K}$$

3. **Map to predicate:**

$map_p(d) = p(map_f(v_1), \dots, map_f(v_n))$ , iff  $p$  is a n-array predicate in  $\mathcal{R}$ ,  
("predicate",  $name_{pred}(p)$ )  $\in d$ ,  $\forall i \in \{1..n\} (name_{pred-atr}(p, i), v_i) \in d$   
 $map_p(d) = nil_p$ , otherwise

# Theoretical Foundation

---

## Mapping into PDDL

e.g.  $map_p(\{("predicate", "at"), ("object", "cup"), ("room", "kitchen")\})$   
 $= at(map_f("cup"), map_f("kitchen")) = at(cup, kitchen).$

1. **Predicate symbols  $\mathcal{R}$ , Function symbols  $\mathcal{F}$**
2. **Name mapping:**

$$name_{pred} : \mathcal{R} \rightarrow \Sigma^* \text{ and } name_{func} : \mathcal{F} \rightarrow \Sigma^*$$

$$name_{pred-atr} : \mathcal{R} \times \mathbb{N} \rightarrow \mathcal{K} \text{ and } name_{func-atr} : \mathcal{F} \times \mathbb{N} \rightarrow \mathcal{K}$$

3. **Map to predicate**
4. **Map to function term:**

$$map_f(v) = v, v \in \mathcal{V}_0$$

$$map_f(d) = f(map_f(v_1), \dots, map_f(v_n)), \text{ iff } f \text{ is a n-array function in } \mathcal{F},$$

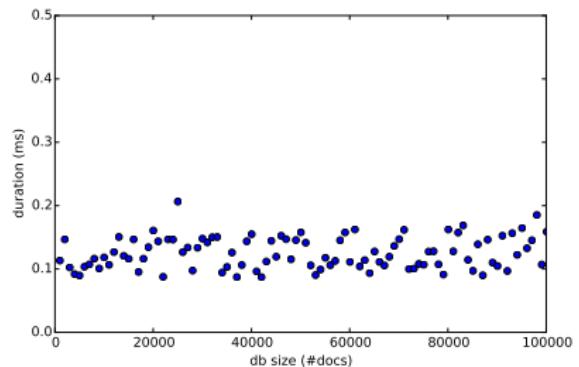
$$("function", name_{func}(f)) \in d, \forall i \in \{1..n\} (name_{func-atr}(f, i), v_i) \in d$$

$$map_f(d) = nil_f, \text{ otherwise}$$

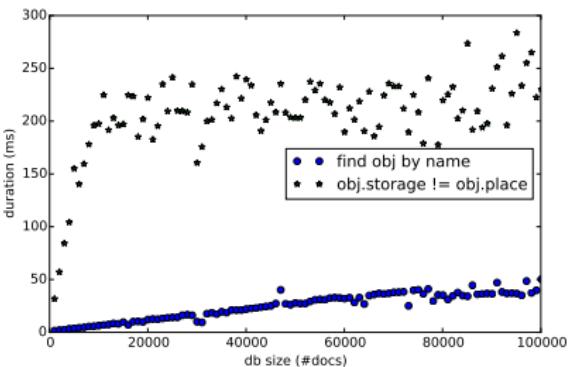
# Trigger/Oplog document

```
{  
    ns: "syncedrobmemo.clipswm",  
    ts: Timestamp(1485369072, 5),  
    op: "i",  
    o: {  
        _id : ObjectId("58c14a14"),  
        relation: "order",  
        id: 2,  
        complexity: "C0",  
        delivery-gate: 3,  
        quantity-requested: 1,  
        quantity-delivered: 0,  
        begin: 209,  
        end: 282 }  
}
```

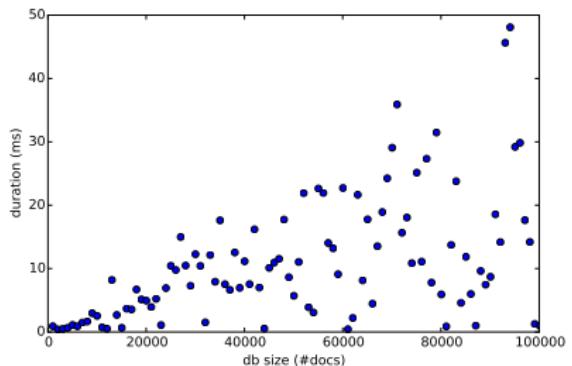
# Evaluation: Duration without Indexing



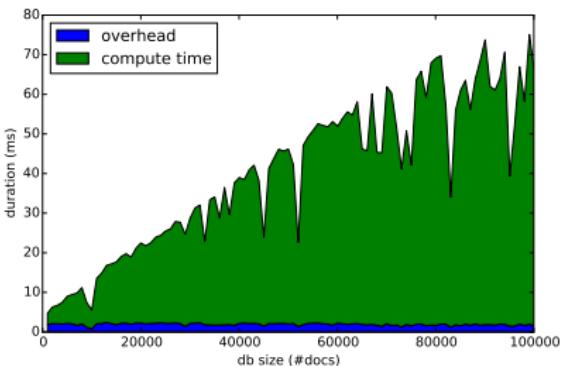
(a) Insertions



(b) Queries

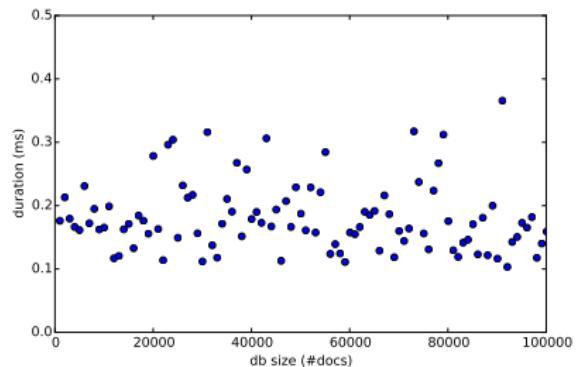


(c) Updates

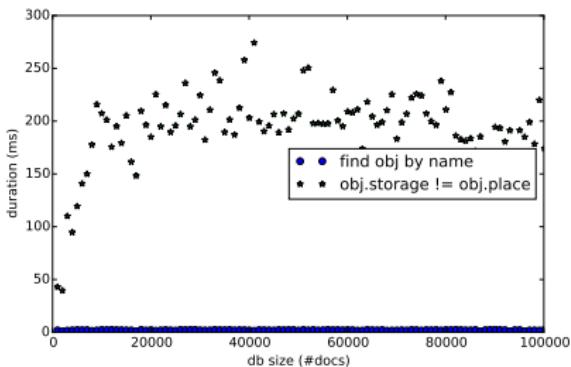


(d) Computables

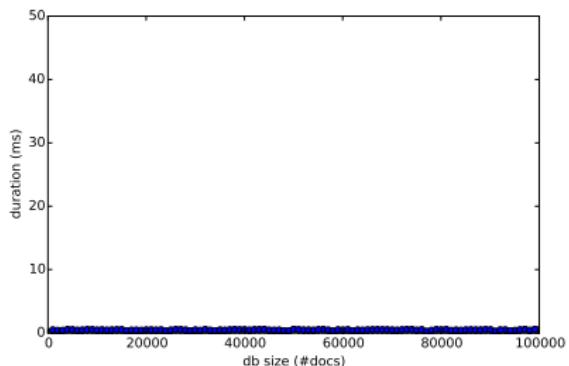
# Evaluation: Durations with Indexing



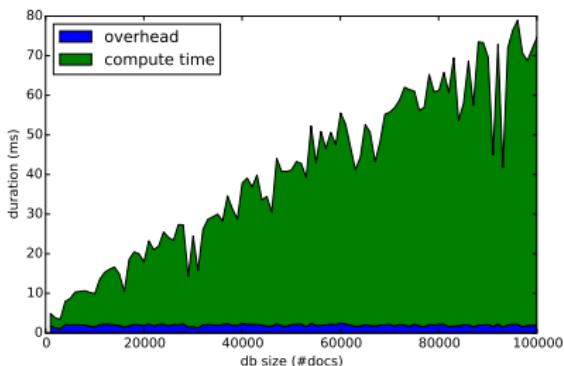
(a) Insertions



(b) Queries

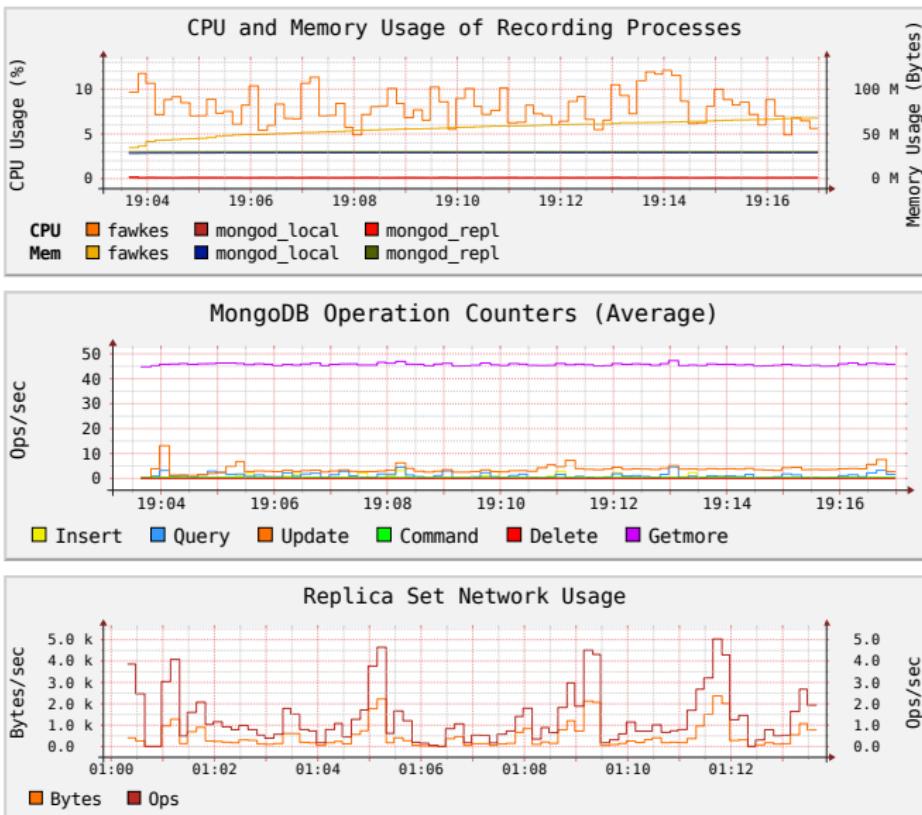


(c) Updates

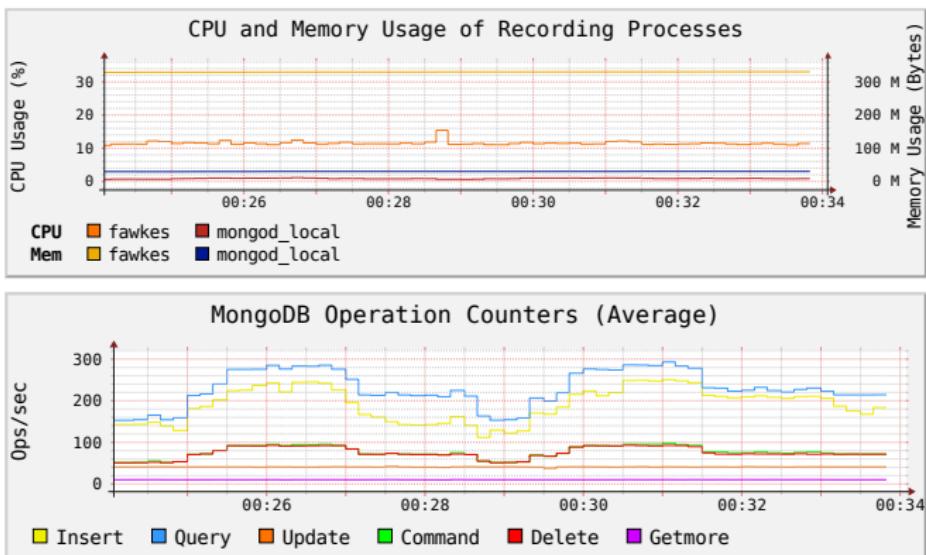


(d) Computables

# Evaluation: Benchmarks RCLL



# Evaluation: Benchmarks Blocks World



# Fawkes

# Fawkes

- Robot Software Framework
  - Component-based software design
  - Hybrid blackboard communication infrastructure  
with specific interfaces/messages
- ⇒ Robot Memory available as Fawkes plugin