

## 1. Introduction

- + Overview
- + Problem Formulation & Motivation
- + Research Questions
- + Methodology & Contributions

RQ1: How can we model the behavior of reactive load balancing to understand the limit in distributed memory systems?

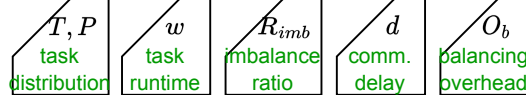
RQ2: How can we proactively balance the load of task parallel applications at runtime?

SQ1: How can we predict the load of tasks to support proactive load balancing at runtime?

SQ2: How can we proactively co-schedule tasks to balance the load?

## 3. Performance Modeling

+ Input:



+ Model:  $F(T, P, w, R_{imb}, d, O_b)$

## 5. Proof of Concepts

+ Simulator: *to evaluate the model*

## 6. Evaluation

+ Synthetic microbenchmarks: MxM, Nbody, ...

+ Real use case: Sam(oa)<sup>2</sup>

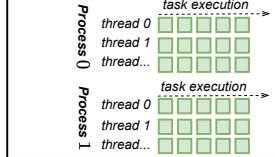
## 2. From Work Stealing to Reactive Load Balancing

### + Preliminaries

+ **Instrument:**  
Task-based Programming Models

Distributed Memory Systems

+ **Object:** Task Parallel Apps



+ Related Work

+ **Constraints:**

Performance Slowdown

Communication Overhead

+ **Target:** Load Balancing (Speedup, Makespan)

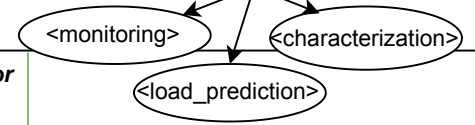
## 4. Proactive Load Balancing (Proactive LB)

A proposed scheme for proactive LB approach

**Main Execution Threads**  
Dedicated Assistant Thread

executing tasks    executing tasks

modularized by components



Three strategies for proactive LB

Feedback Task Offloading

ML-based Task Offloading

- Stage 1: Task characterization
- Stage 2: Online load prediction
- Stage 3: Proactive Task Offloading

Co-scheduling Tasks across Multiple Applications

+ **C++** Implementation: plugin tool upon *Chameleon*

+ **Python** for visualizing and checking prediction models

## 7. Conclusions & Future Work

+ Conclusion

+ Extended from the direction of performance modeling

+ Extended from the direction of proactive load balancing