alsched: Algebraic Scheduling of Mixed Workloads in Heterogeneous Clouds

Alexey Tumanov

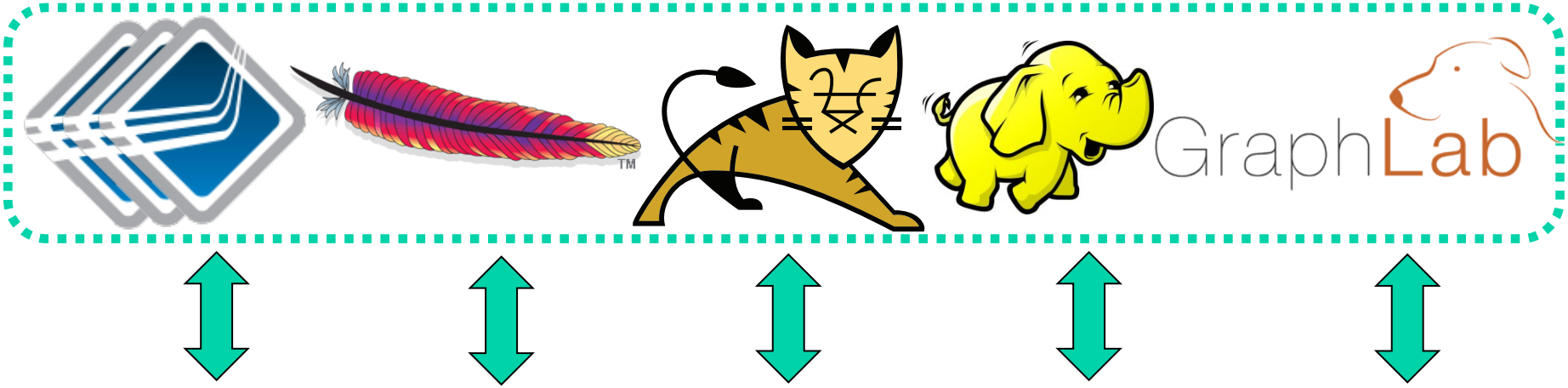
James Cipar, Michael Kozuch, Greg Ganger

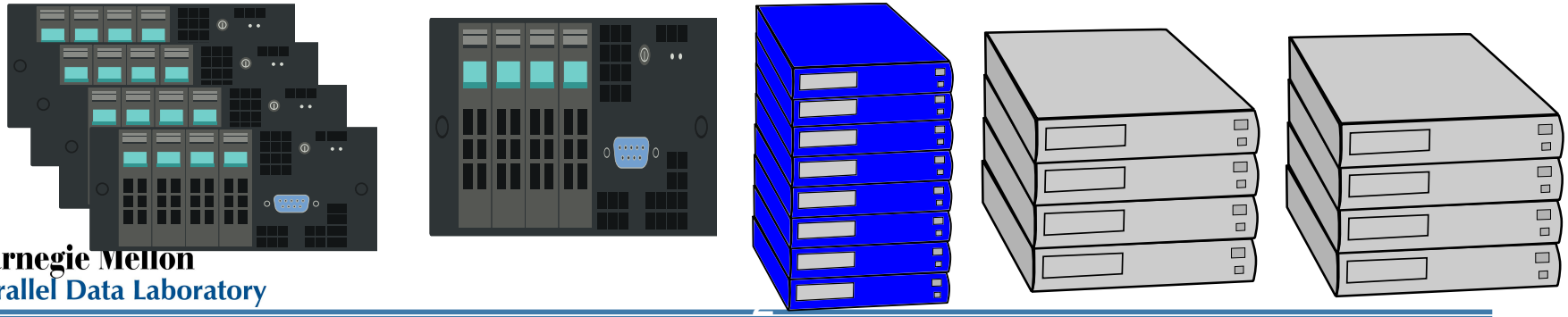
PARALLEL DATA LABORATORY

Carnegie Mellon University

Context & motivation: heterogeneity

- HW & SW diversity → new scheduling challenges



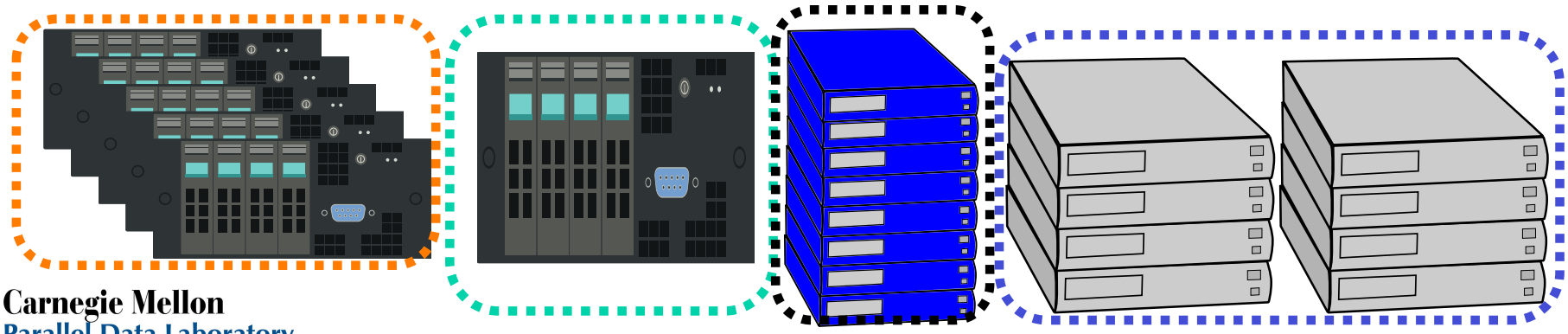


Key takeaways

- "Soft" scheduling constraints are important
  - placement preferences and [multi-level] fallbacks
- Need a way to quantify them
  - to enable better placement decisions
- Promising approach: composable utility functions

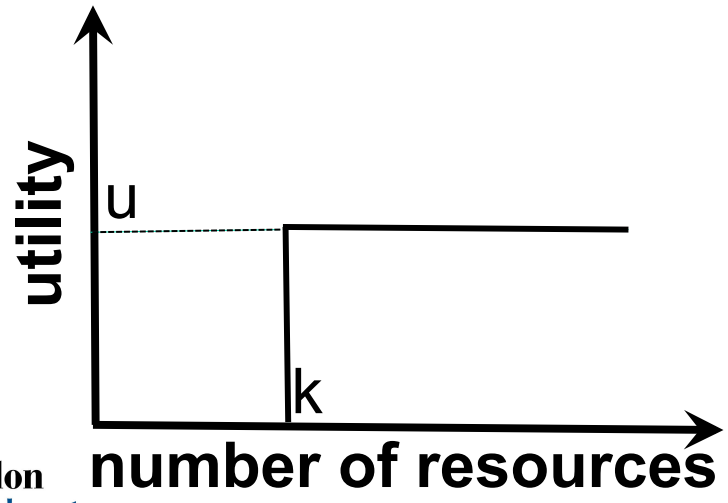
Soft constraint specification

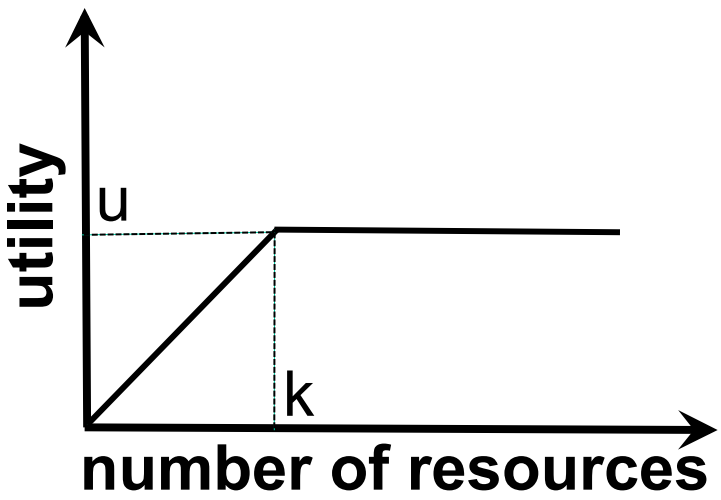
- Utility function u(P): placement P → utility u
  - expensive to specify over all resource subsets
  - specified over equivalence classes
- Equivalence class – a set of interchangeable resources from user's perspective
  - e.g., machines with a GPU, same rack machines



Utility function primitives

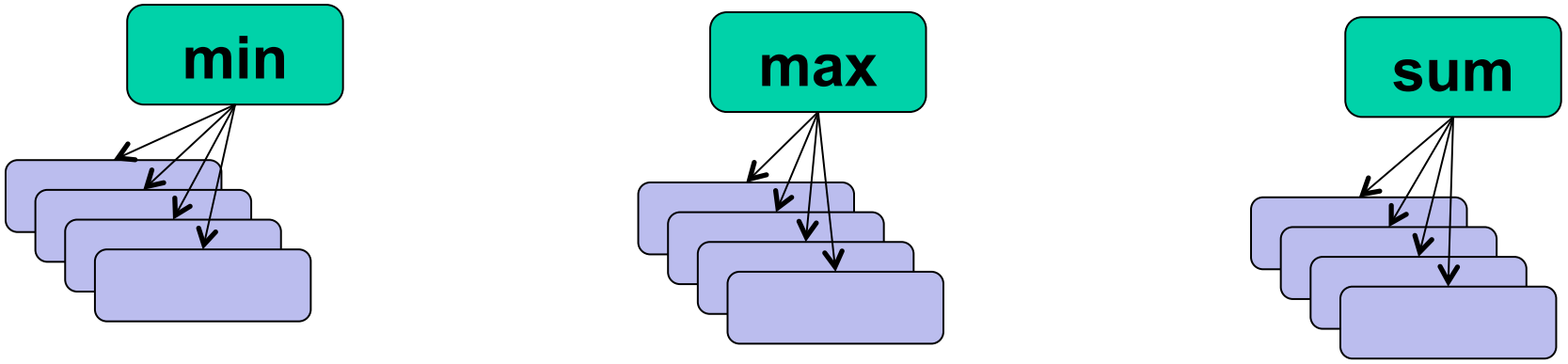
- Express utility as a function of primitives
- `n Choose k'
  - maps a specified quantity **k** from the **n** machines of the given equivalence class to scalar utility value **u**
- Linear `n Choose k'
  - linear increase in utility up to k





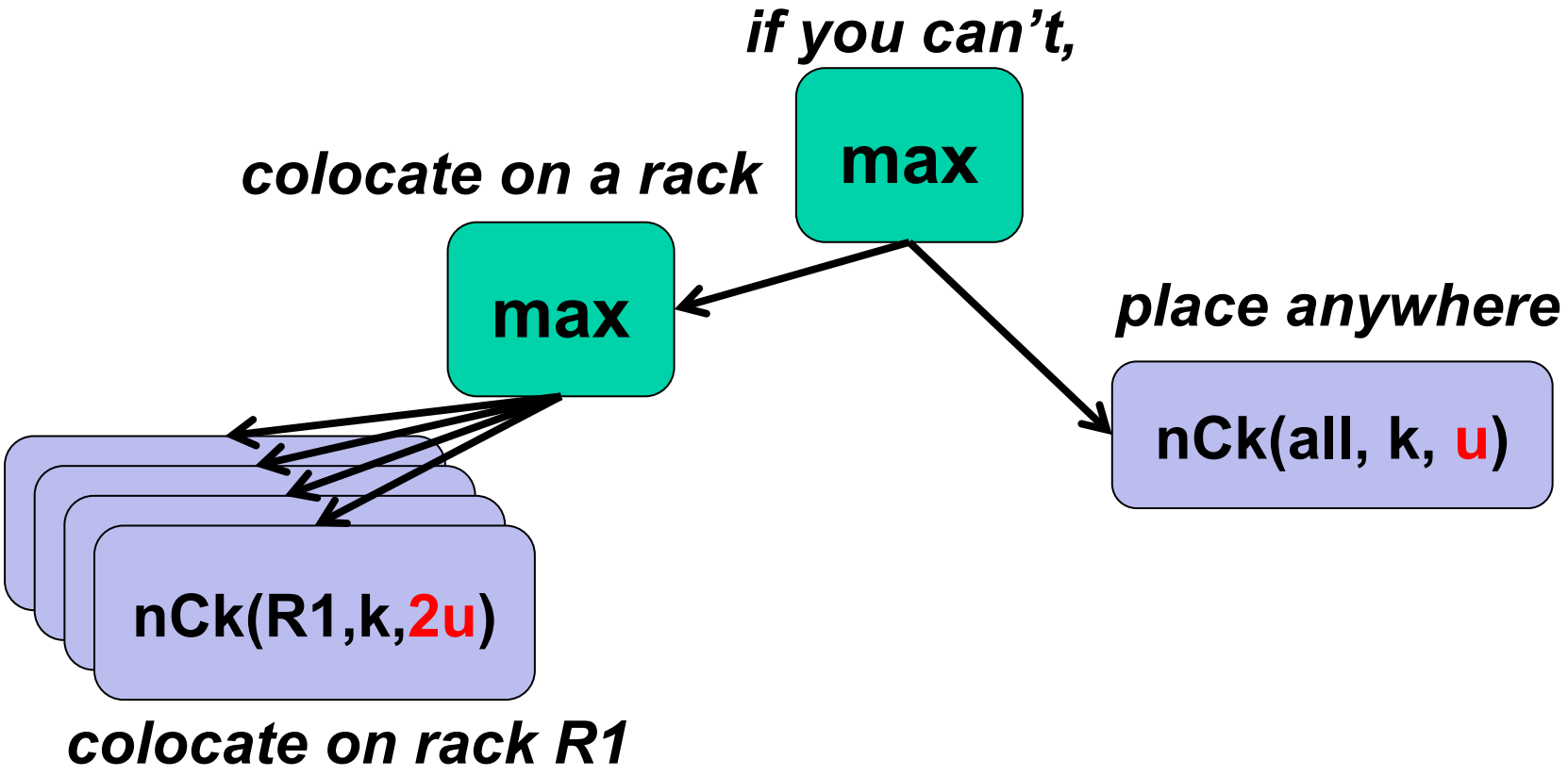
Composition operators

- Primitives are composed with operators
- Min(u_1,u_2,… u_n) → evaluates to minimal u_i
- Max(u_1, u_2, … u_n) → evaluates to maximum u_i
- Sum(u_1, u_2, … u_n)
- Others (e.g., scale, barrier)



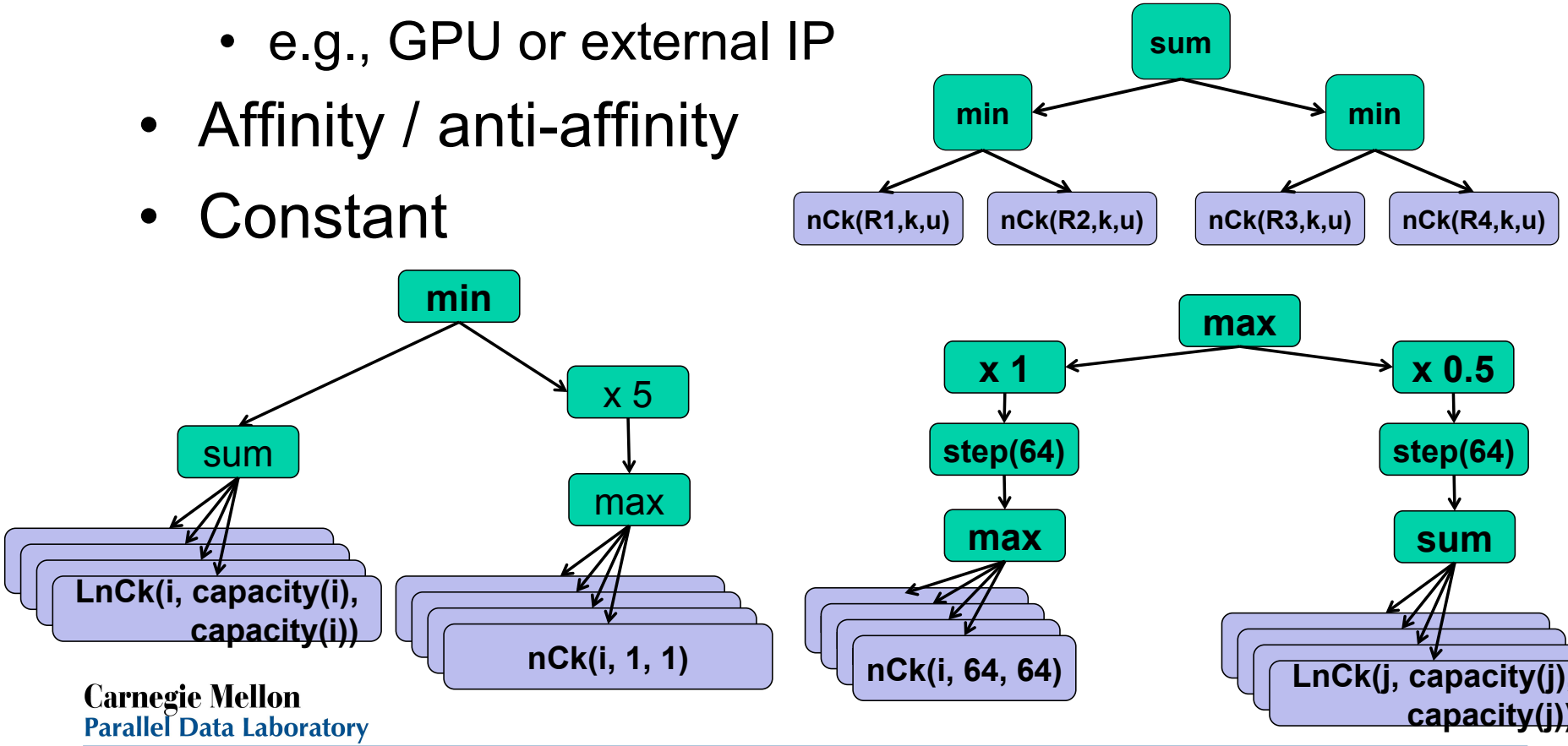
Example: locality

- Colocate k tasks on the same rack, else schedule them anywhere



Many other examples

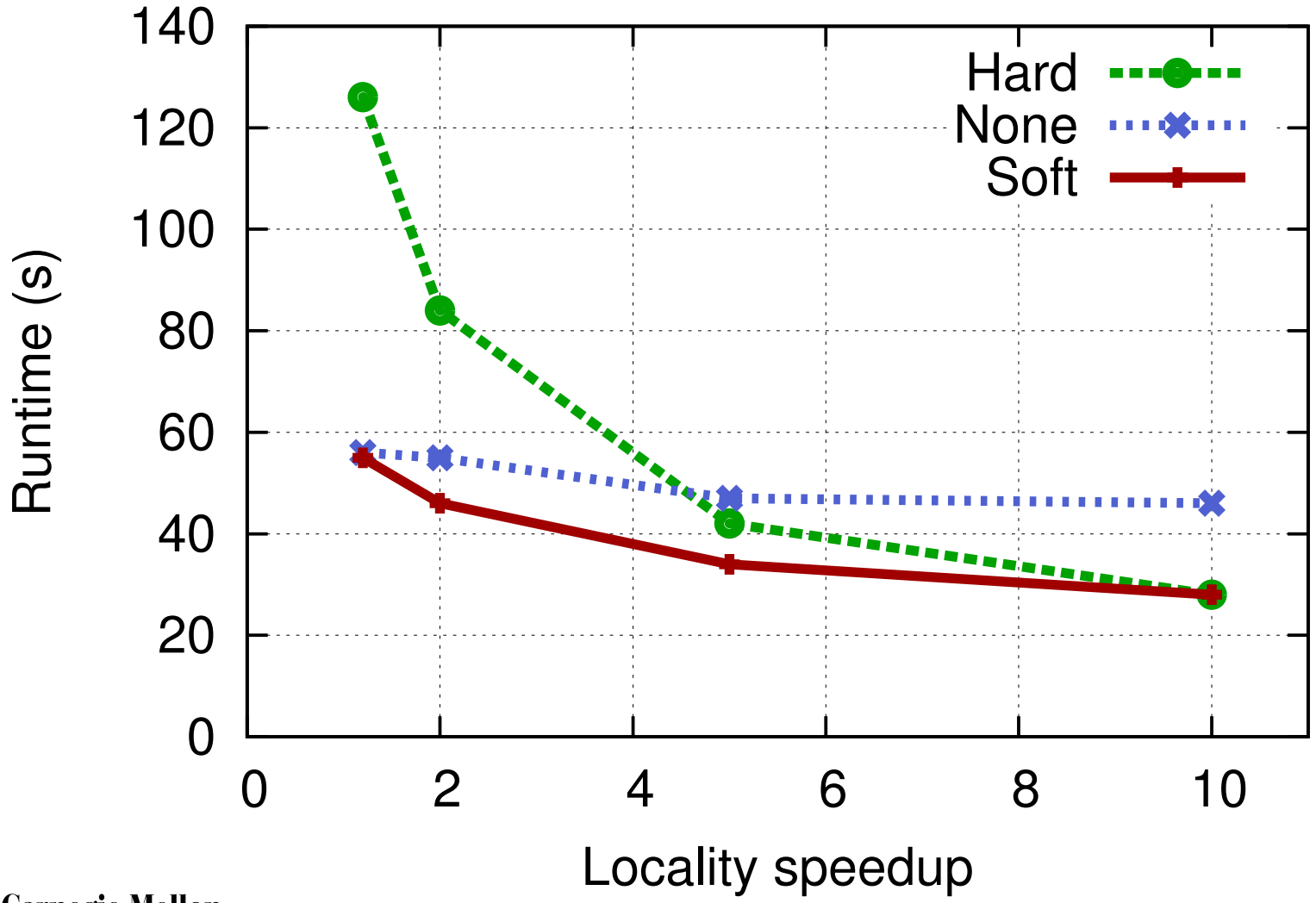
- Primary + backup service instance
- Machines with a certain resource of interest
  - e.g., GPU or external IP
- Affinity / anti-affinity
- Constant



One initial experiment

- Simulated n-body type workload
- Three scheduling policies compared:
  - Soft – soft constraint aware placement
  - Hard – soft constraints are treated as hard
  - None – soft constraints are ignored
- Variable: locality speedup
- Metric: total runtime to finish all work

Soft constraints matter



Key takeaways

- Soft constraints are important
  - Need to quantify them explicitly
- Composable utility functions are promising
  - flexible, simple, expressive

Ongoing research

- Automatic generation of utility functions
- Handling imperfectly specified utility functions
- Handling placement change decisions
  - interactions for reclaiming allocated resources

Conclusion

- Soft constraints are important
  - Need to quantify them explicitly
- Composable utility functions are promising
  - flexible, simple, expressive

Alexey Tumanov, James Cipar, Michael Kozuch, Greg Ganger, "***alsched**: algebraic scheduling of mixed workloads in heterogeneous clouds*". SoCC'12.