Today’s parallel software is written for yesterday’s systems, dominated by slowest resource, and optimized for a single scale. This leads to under-utilization and resource wastage.

Multi-scale resource ensembles

Ensembles are collections of resources:
- Internally homogeneous
- Resources at same scale of parallelism
- Independent scheduling units

Self-characterization
- Declarative description
  - CPUs, RAM, NICs/ICs, Disks, GPGPs

- Static profiling
  - Micro-benchmarks
  - Pairwise profiling
  - I/O profiling

- Continuous profiling
  - Benchmarking tasks

Structure: relating ensembles

Nesting (A ⊆ B ⊆ C)

Peering (C-D)

Digraph

Hierarchy

Example

Interaction
- Exchange state updates
  - Spawn rate
  - Completion rate
  - Utilization
  - Data held
  - Ensemble self-description
- Model as a flow network
  - Each ensemble is a source, each resource a sink.

Task migration

Benefits
- Faster job completion
- Scale-adaption
- Higher resource utilization

Preliminary results

Synthetic workload simulation
- Inter-arrival time: \( \text{Exp}(\lambda_2) \)
- Task duration: \( \text{ln}^2(N(\mu, \sigma^2)) \)
- Job size: \( \text{Exp}(\lambda_1) \)

Avoid fan-in contention
- Gossip current spawn/completion rates and queue lengths to control task flow

Assess benefit of task migration
- Performance modelling, extrapolation from historic measurements

Controlling task granularity
- Compiler pass (LLVM) to generate tasks at multiple scales from single program

Open questions and challenges

- Performance heterogeneity
- Scales of parallelism
- Resource heterogeneity
- Multi-scale resource ensembles

Malte Schwarzkopf
Steven Hand

http://www.cl.cam.ac.uk/~ms705/

Email: Malte.Schwarzkopf@cl.cam.ac.uk