Getting Your Priorities Right

Fine-grained threading
Computation
E.g. early compress

Interaction
E.g. GUI

E-mail client

Need priorities - Millions of threads, no way to distinguish

2 problems w/ most ways of handling priorities
1. Fixed order → Anti-modular
2. Priority inversions - high-prio thread waiting for low-prio

Solution: PriML
Partially ordered priors - prior order декл
spawn/sync + priorities
Q sort code

Type sys. prevents prior inversions

\[ c := \cdots | c \text{ thread } [p] | c \text{ cmd } [p] | \forall \pi : C . c \]
\[ p := p \mid \pi \]
\[ e := \cdots | \Delta \pi : C . e \]
\[ C := p \leq p \mid C \land C \]
\[ m := x \leftarrow e ; m \mid \text{spawn } [p] ; \pi ] \mid m ; \mid \text{sync } e ] \mid \text{ret } e \]
\[ \Gamma \vdash m \div 2 \mathbin{@} p : m \mathbin{\text{returns}} a, \mathbb{r} \mathbin{\text{uns}} \mathbin{@} p \mathbin{\text{rio}}. \ p \]

\[ \frac{\Gamma \vdash m \div 2 \mathbin{@} p'} {\Gamma \vdash \text{spawn}(p'; 2) \mathbin{\text{\&}} 2 \mathbin{\text{thread}}(p) \mathbin{@} p} \quad \text{(SPAWN)} \]

\[ \frac{\Gamma \vdash e : 2 \mathbin{\text{thread}}(p) \quad \Gamma \vdash p \mathbin{\leq} p'} {\Gamma \vdash \text{sync } e \div 2 \mathbin{@} p} \quad \text{(SYNC)} \]

\[ \frac{\Gamma, C \vdash e : 2} {\Gamma \vdash \lambda \mu. C. e : \text{Var}. C. 2} \]

**Cost Semantics**

Model program as a DAG - label threads w/priorities

- **Greedy schedule** - Assign vertices to procs so no procs are idle unless necessary.
  \[ T \leq \frac{w \cdot p}{s} \]
- **Prompt schedule** - Greedy + run highest prio vertices possible

Bound response time: How long from when a thread is spawned until it completes.

Let \( a = s \ldots t \)

\[ RT(a) \leq \frac{w_p (w)}{p} + s \cdot a \cdot \pi \]

\( w_p (w) \) - competitor work, \( a \cdot \pi \) - \( a \)-span (longest path ending at \( t \))
Bound holds unless there is a prior inversion.

Cost semantics: \( M \vdash v : g \)

**Theorem:** If \( \tau : m \vdash c @ p \) and \( M \vdash v : g \), then \( g \) does not have a prior inversion.

Dynamic semantics: \( m \Rightarrow m' \)

- thread pool
- \( a \Leftrightarrow m \)

**Theorem:** If \( \tau : m \vdash c @ p \) and \( a @ m \Rightarrow^* m' \) and \( m' \) final and \( a \) thread \( a \) is active for \( T \) transitions. Then \( M \vdash v : g \) and \( \exists \) a prompt schedule of \( g \) in which \( RT(a) = T \).