Wrapping up

Looking back

- Check out of rooms
- Kenichi on challenge (Albert on OpenCV)

Looking forward

- Task list from Jeremy's notes last night
- Group blog
- Next StageHPC
- Please leave your parting message!
- Bus to train station departs at 13:45 downhill

What to advertise (both existing and new), 1/2

solve HPC problems

- respond to and improve Kenichi's list "Shonan Challenge"
- Matthew's list, Reiji's infinite stream
- from kernels to larger libraries and (one-off) applications
- raising abstraction level (avoid $O(nm)$) across computation models (parallel processing, FPGA, GPU)
- share patterns (intermediate representations, genetic algorithms) across domains, at this kind of event
- stencil programs, applied math
- data layout optimization

What to advertise (both existing and new), 2/2

specialize code in Fortran, restricted Python, whatever people use

- consider existing infrastructure: Tempo, Glück's Fortran specializer, PGG (easy to retarget for another output language), ROSE, Scala LMS, MetaOCaml
- pursue and control partial evaluation

bring/keep MetaOCaml up to date

- native code compiler for BER MetaOCaml
- interest from people and support from OCaml team
- consortium funding for a research programmer

need rewriting around staging
static safety and run are desirable but not absolutely necessary

How to advertise

audiences

- users (e.g., ICCS), grad students, scientists
- experts (e.g., SC)

venues

- domain venues (biology, astrophysics, vision, linguistics, . . .) will pay attention to impact
- tutorials
  - ICFP (but that's not where customers are)
  - SC! ICCS! They are two different audiences
  - HIPEAC (next tutorial deadline June 1, 2012)
- SC/ICCS exhibit/BOF
- SIAM mini-symposia
- Web bibliography of staging applications and research
- annual StageHPC
- state-of-the-art document
- ecosystem of (domain-specific) offerings, branded "staging", starting with gateway drug such as operator overloading

Vision (draft challenge abstract, plagiarizing from POPLmark paper at TPHOLs)

How close are we to a world where

- every paper on high-performance computing is accompanied by an electronic appendix with machine program generators?
- natural-science grad students no longer need to translate their high-level formulas into Fortran?

We propose an initial set of benchmarks for measuring progress in this area. Based on the metatheory of System F¡:, a typed lambda-calculus with second-order polymorphism, subtyping, and records, these benchmarks embody many aspects of programming languages that are challenging to formalize: variable binding at both the term and type levels, syntactic forms with variable numbers of components (including binders), and proofs demanding complex induction principles. We hope that these benchmarks will help clarify the current state of the art, provide a basis for comparing competing technologies, and motivate further research.