Mega-Scale Computing

Based on Low-Power Technology and Workload Modeling

MEGASCALE

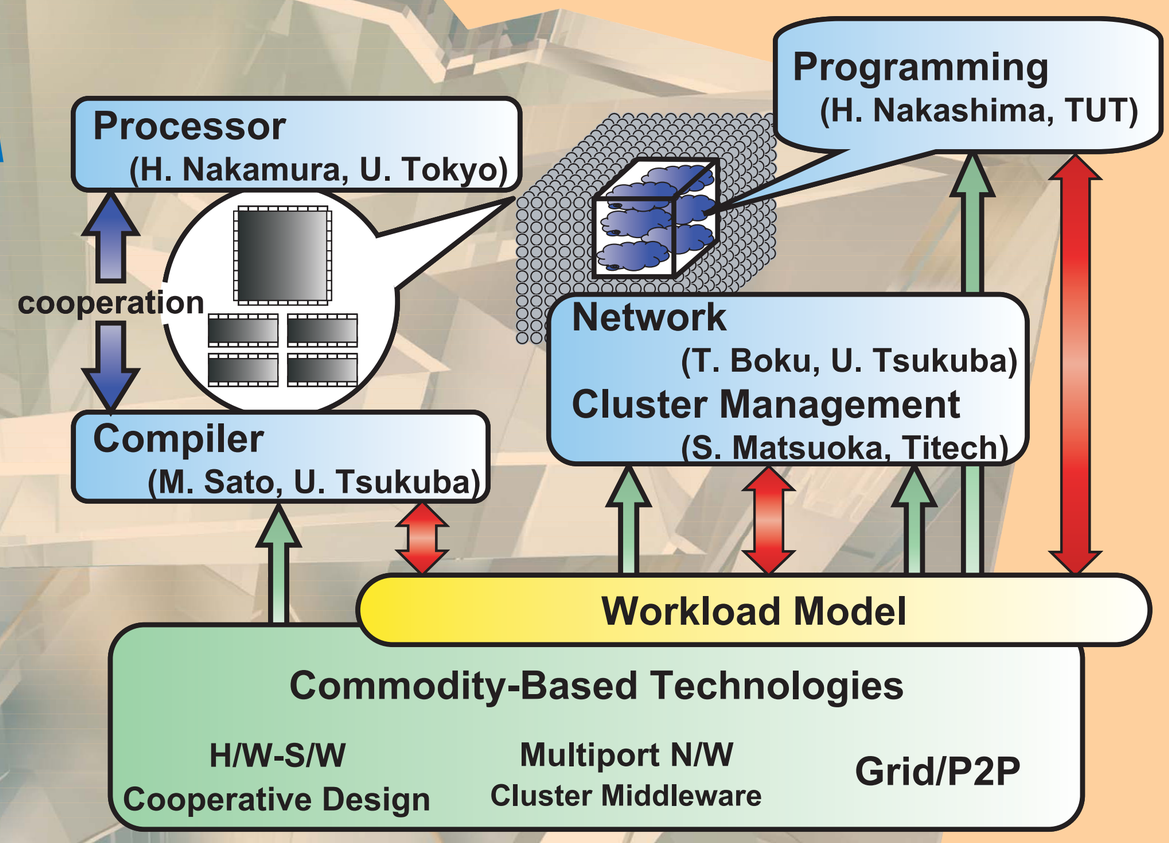
Introduction

Peta-FLOPS computing is eagerly desired in wide area of applications including genomics, life/bio-science, huge-scale simulation, and other computational science fields. Our fundamental question is “Can we achieve Peta-FLOPS by enlarging MPP or clusters?” and our answer is “NO” because it requires an infeasibly huge space and power budget. Our answer to the question “How can we achieve it?” is *mega-scale computing* in which millions of processors are clustered by *commodity-based* technologies. Mega-scale computing will be *feasible* because our low-power architecture cooperating with our power-aware compiler makes it possible to package the system in a feasible space with a feasible power consumption. It will be also *dependable* owing to our highly reliable networking and system-level dependability management guided by the modeled behavior of workloads. The modeling is also the key technology in our multi-tier programming language and system by

Research Overview

Our research, supported by a JST/CREST program entitled “Advanced Information Technology for Future C&C Society and Community”, aims to study key technologies for our ultimate goal, *mega-scale computing*, and to integrate them on our 10^3-scale prototype named *MegaProto*. Each technology is studied by one of the following research groups.

- ► Processor group is designing a new low-power and high-performance processor architecture named *SCIMA* that has an on-chip software controllable memory.
- ► Compiler group aims to have compilers for *SCIMA* and *MegaProto* to generate power-aware optimized codes exploiting memory hierarchy.
- ► Network group is developing a reliable and high-performance network named *RI2N* based on commodity multi-port networking technology.
- ► Cluster management group works on *dependability* issues including soft-failure detection and optimized checkpointing guided by the model of workloads.
- ► Programming technology group is developing a script parallel programming language *MegaScript* for easy-to-write multi-tier mega-scale programming.



MegaScript Language

Two-tier task-parallel programming

- ► lower = existing sequential/parallel program (task)
- ► upper = task network for parameter survey, optimum solution search, etc.

Easy-to-write script language

- ► object oriented based on Ruby
- ► task objects communicate through stream objects connected to standard I/O
- ► hierarchical modules
- ► application oriented interface for end-users
- ► customizable runtime modules for heavy-users

Meta-programming for workload modeling

- ► describe task model as a program
- ► freely abstracted/detailed to represent programmer’s knowledge for task scheduling/allocation
- ► model instantiation with runtime info (e.g. args)
- ► model refinement with runtime profile