Mega-Scale Computing
Based on Low-Power Technology and Workload Modeling
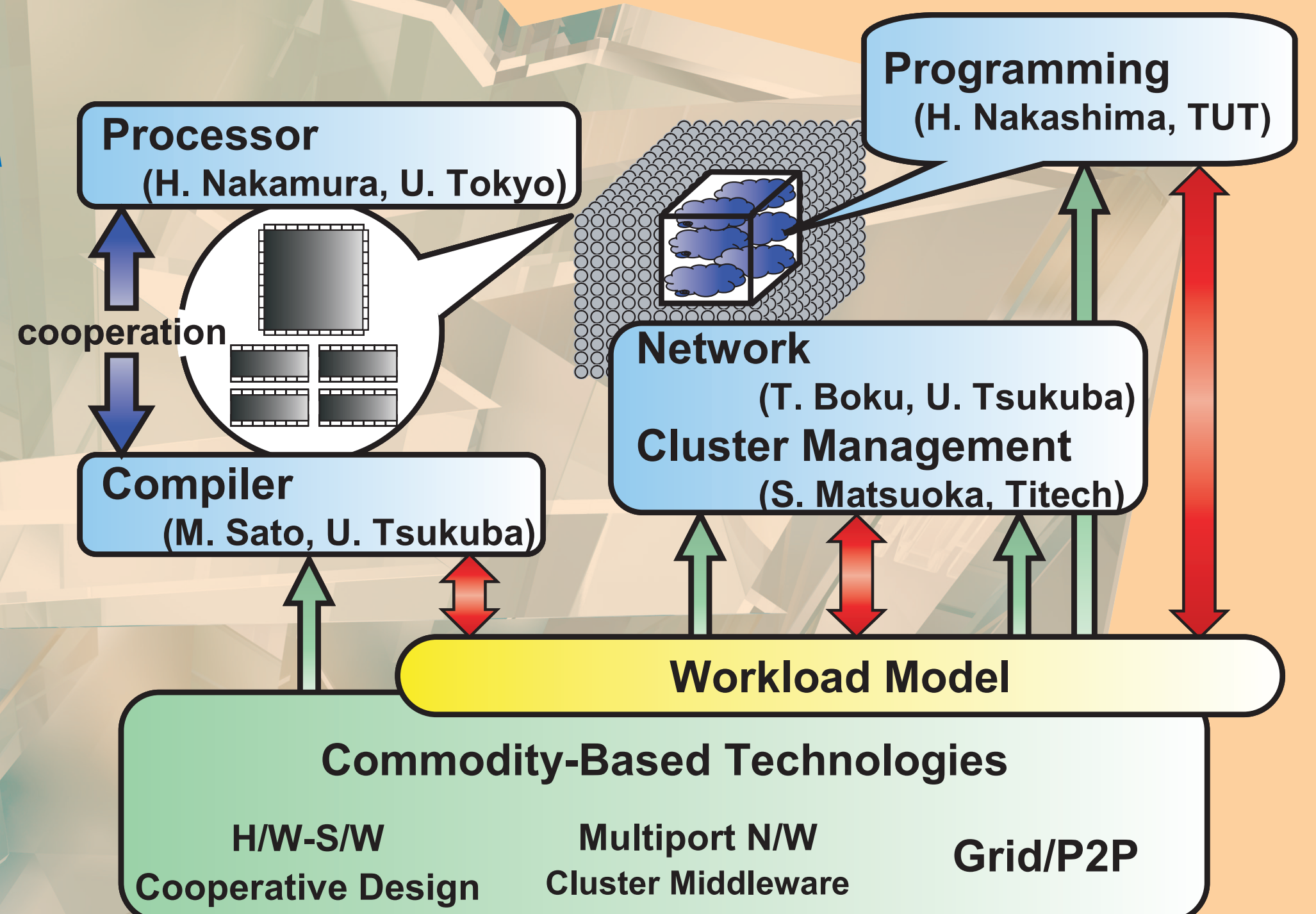
MEGASCALE

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 which a huge mega-scale problem is *programmable*.

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.

Processor (H. Nakamura, U. Tokyo)

Programming (H. Nakashima, TUT)

Network (T. Boku, U. Tsukuba)

Cluster Management (S. Matsuoka, Titech)

cooperation

Compiler (M. Sato, U. Tsukuba)

Workload Model

Commodity-Based Technologies

H/W-S/W Cooperative Design   Multiport N/W Cluster Middleware   Grid/P2P

MegaScript Language

► Two-tier task-parallel programming
► Easy-to-write script language
► Meta-programming for *workload modeling*
► *Model Refinement* with runtime profile

real   initial model   refined model

exec time

model error

MegaScript Program

```
p BranchAndBound.exec(
    HugeSim,[10000,...])

class HugeSim < Task
  def initialize(a)
    @arg=a
    @exefile='huge-sim'
  end
  def behavior
    FOR i in 1..@arg[1]
      FOR i
        compute();
      END
    END
  end
end
```

object

meta

MegaScript Runtime

B&B	GA	Param Surv
M/W	2Dmesh	Ring
Sched	Conn	Task Mgr
Task	Stream	Task Alloc

Workload Model

Profiling

Refinement

FT

(sec) / (%)

MG

IS

(nodes)