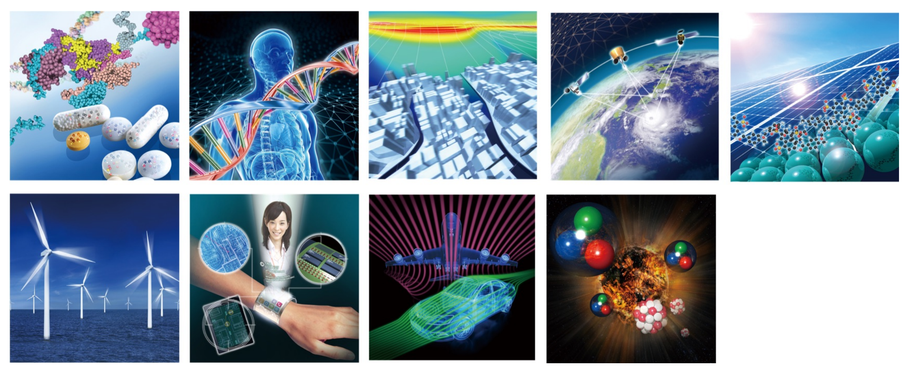
Codesign Supercomputer Fugaku and Target Applications through Performance Optimization

Kazunori Mikami, Hirofumi Tomita, Kazuo Mikami
RIKEN Center for Computational Science

Supercomputer Fugaku
- the flagship supercomputer of Japan
- development started in FY2014, due production service in FY2020



Target applications
- represent the social/scientific priority issues selected by the ministry
- use different numerical schemes, grid structures, data types
- cover typical applications workload in total

Codesign goal
- complete Fugaku making it deliver the efficient computing capability for a range of applications
- modernize target applications to achieve expected performance on Fugaku, and on similar HPC systems

Target application	Type of algorithm	#nodes /job	structured grid	unstructured grid	particle method	dense matrix (local)	dense matrix (dist.)	comm. latency	comm. B/W	neighbor comm.	global comm.	heavy I/O
GENESIS	Molecular dynamics for proteins	1			☑					☑	☑	
Genomon	Genome data analysis	96										☑
GAMERA	Earthquake simulation using hybrid grid FEM	entire		☑				☑	☑		☑	
NICAM+LETKF	Weather prediction using structured FVM + LETKF	131072	☑			☑			☑	☑		☑
NTChem	Electron correlation energy using molecular orbital method	20732				☑	☑		☑	☑	☑	
ADVENTURE	Structure analysis FEM	4096		☑		☑	☑	☑		☑	☑	
RSDFT	Electronic-structure calculations using density functional theory	10368	☑			☑			☑	☑	☑	
FFB	Thermal fluid flow using FEM	entire		☑						☑	☑	
LQCD	Lattice QCD simulation using grid Monte Carlo method	147456	☑					☑	☑	☑	☑	

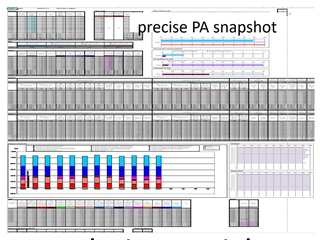
Codesign approach
- simultaneous development of the system and applications using performance measures, under power constraint

Codesign from target application's view

Repeat optimizing the application, interacting with system development
- identify the performance bottleneck utilizing performance tools
  - optimization effort in applications
    - alternative algorithm, source level manual tuning
  - system design improvement request for feasibility study
- rerun the cycle with new source code and new design parameters
  - look for further optimization possibility from different angles

Performance bottleneck analysis using tools

tools for analysis on existing platform
- general profilers(fipp/fapp) statistics
- precise PA (HWPC) for detail component analysis
  - flops, intops, B/W, SIMD rate, cash hit/miss, ..



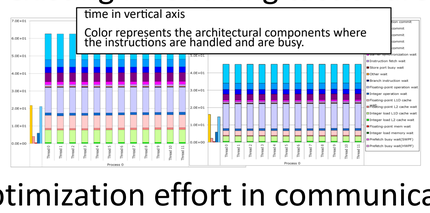
tools for Fugaku performance prediction
- PA prediction tool using FX100 precise PA : static, interpolation, quick
- instruction simulators : dynamic, accurate, expensive
- Fugaku prototype test vehicle
  - prediction tools are private ,except for RIKEN simulator available at https://github.com/RIKEN-RCCS/riken_simulator

Addressing bottlenecks

discuss how the bottleneck should be improved – apps, system S/W, H/W
- applications: tuning effort in applications is essential
- system S/W: enhancing compilers and libraries
- System H/W: specification design and implementation enhancement

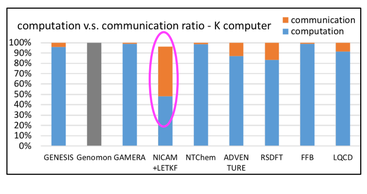
Optimization in applications

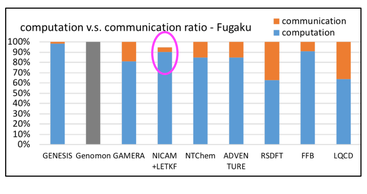
manual tuning is required in many situations
- choose good algorithm, most importantly
- simplify the code structure, rewrite code for better instruction mix
- existing HPC tuning tactics are usually effective



- Scalar code example
  - Loop fusing for longer loop
  - Dimension exchange for shorter memory access stride
- The effect
  - reduced data access wait time (lower green and flesh color)
  - still long f.p. ops wait time (violet) because of sequential(non-SVE) ops

optimization effort in communication bound app
- NICAM+LETKF deployed new frame work adopting global alltoall among all members, instead of sequential gather/scatter from every ensemble member, which resulted in dramatic performance improvement

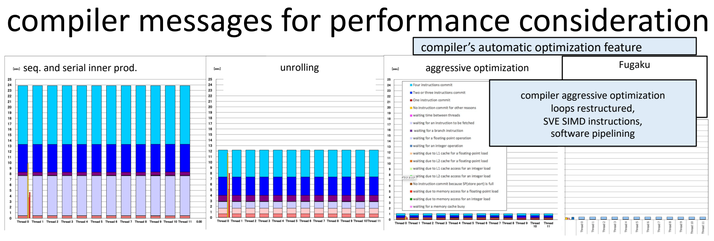




NICAM+LETKF
1024 ensemble members x
128 nodes/member

Optimization in system software

incorporate the tuning capability into compilers as much as possible
- automatic optimization feature, additional fine control directives, useful compiler messages for performance consideration



Optimization in system hardware

enhance CPU architecture to resolve bottleneck indicated by applications
- A64FX instruction handling mechanism in L1 cache
  - combined gather mechanism
    - reduce the gather overhead in 128B address space
  - cache eviction reducing mechanism
    - mitigate cache refill penalty for codes needing lots of load streams

```
do i=1,n
a(i) = b(index(i))
end do
```

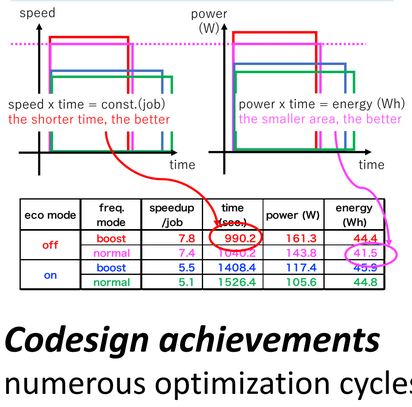
```
do i=1,n
a(i,j) = (b(i,j)+b(i+1,j)+b(i,j+1)) * (c(i,j)+c(i+1,j)+c(i,j+1)) + ..
end do
```

User option in performance and power balance

users can choose the combination of:
- CPU frequency mode (2GHz/2.2GHz) and economy mode (on/off)

administrators can deploy a policy to promote the intended operation

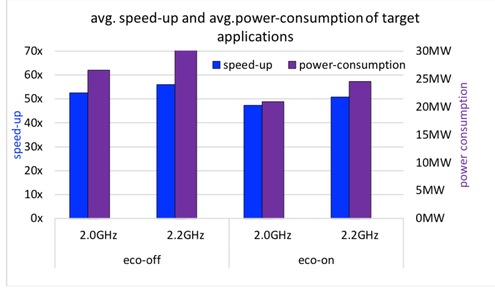
- NTChem power profile example, same job, same # of nodes/job
- speed and power axes are both relative to eco off normal mode



eco mode	freq. mode	speedup /job	time (sec.)	power (W)	energy (Wh)
off	boost	7.8	990.2	161.3	44.4
	normal	7.4	1040.2	143.8	41.5
on	boost	5.5	1408.4	117.4	45.9
	normal	5.1	1526.4	105.6	44.8

- Average of Target applications

The results obtained on the evaluation environment in the trial phase do not guarantee the performance, power and other attributes of the supercomputer Fugaku at the start of its operation.



Codesign achievements

numerous optimization cycles resulting in:
- Target applications production ready from day 1 on Fugaku
  - optimized performance on Fugaku, also efficient on modern HPC systems
  - in depth programming guide for succeeding applications
- Fugaku hardware and software oriented to HPC applications demand
  - 60+ codesign requests were reflected in the system development phase
  - early stage codesign contributes to the systems architecture:
    - processor and memory specification, data mechanism
  - later stage codesign contributes to the system software:
    - compiler optimization feature, numerical library, MPI/network library

codesign	app	design request	design implementation
1	common	separate services from computation	integrated assistant cores
2	LQCD	zero noize operating system	McKernel OS option per job
3	FFB	relieving L1 pressure in indirect SIMD load	combined gather mechanism
4	GAMERA	automatic and optimized loop fission	compiler enhancement
60 more	…	…	…