神戸大学　超並列処理特論　2021年9月16日

計算科学と
スーパーコンピュータ「富岳」

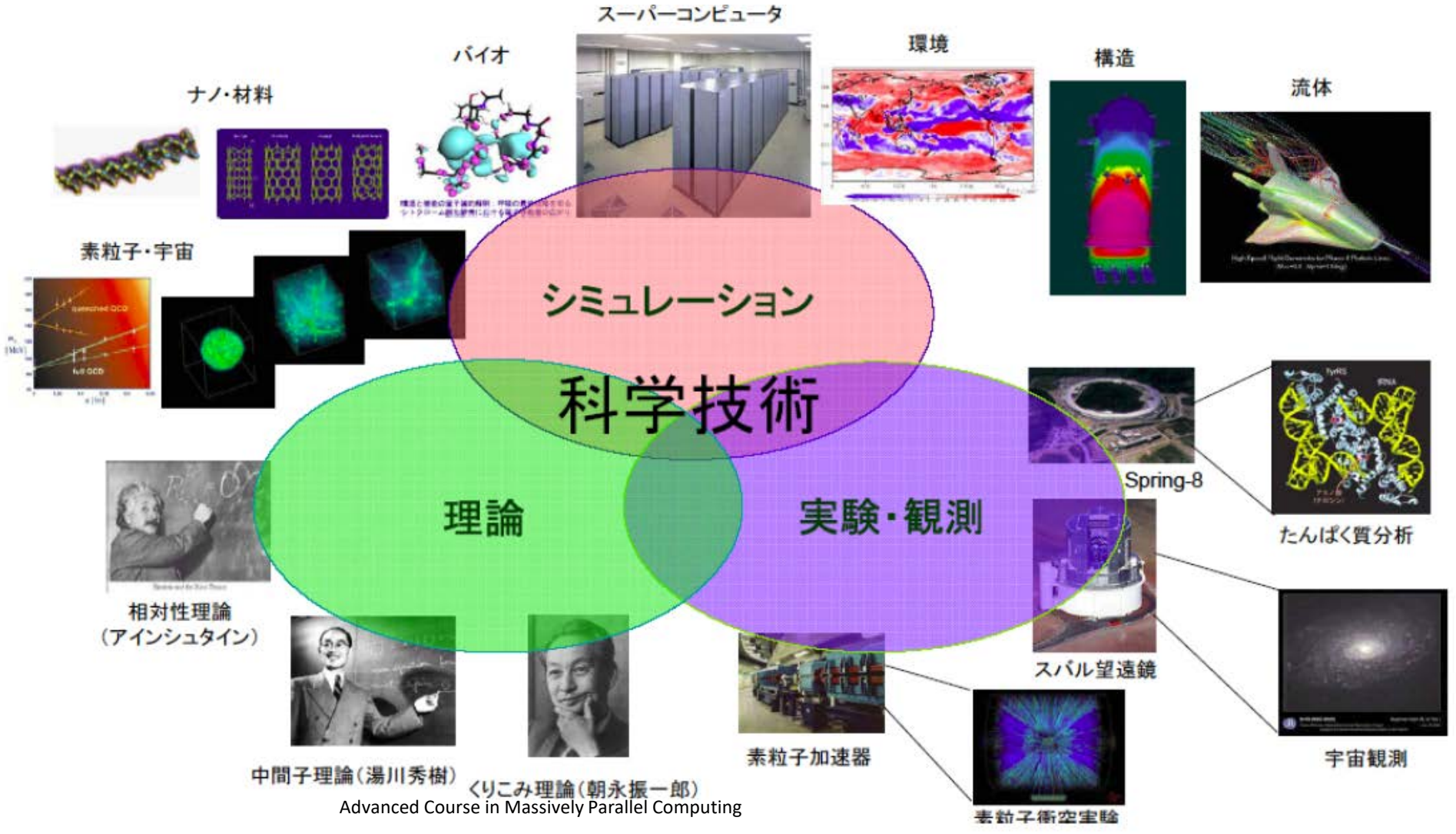
理化学研究所　計算科学研究センター
副センター長
フラッグシップ2020　副プロジェクトリーダー
アーキテクチャ開発チーム・チームリーダー

佐藤　三久

石川　裕　FS2020プロジェクトリーダー

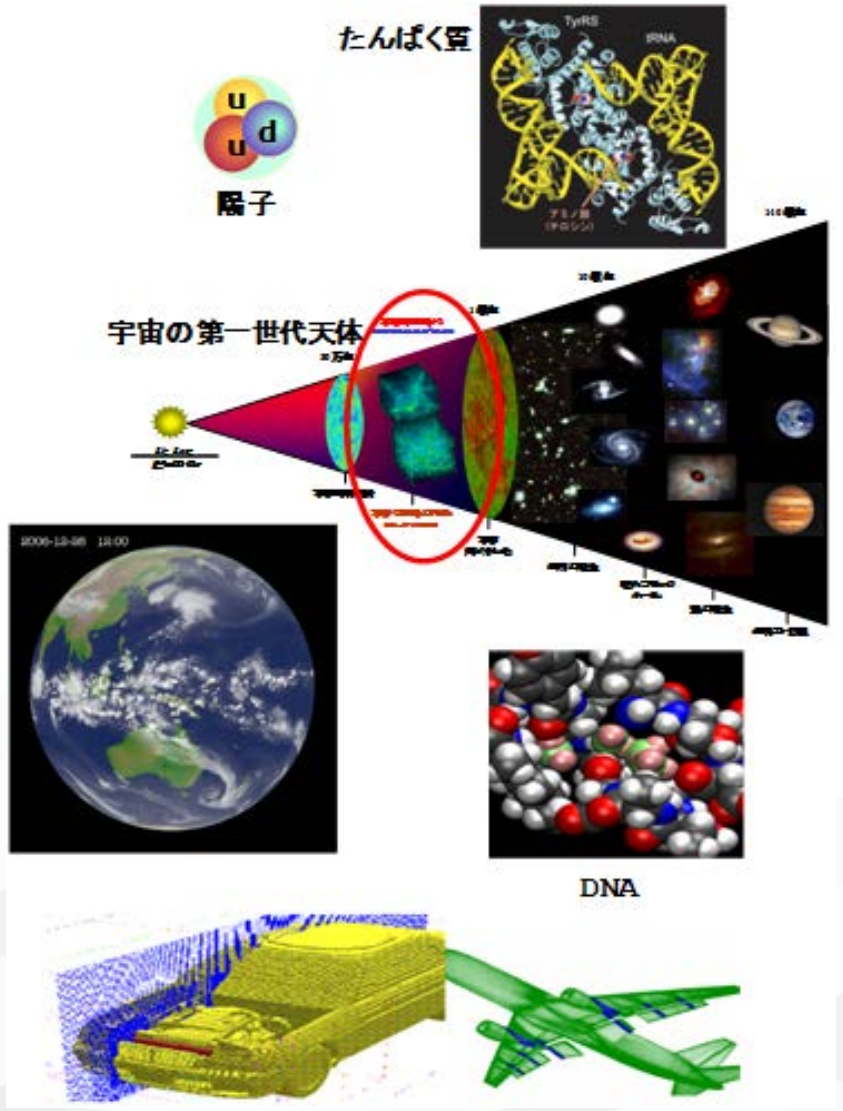
科学の３本柱としての計算科学

- スーパーコンピュータを用いた大規模シミュレーションを中心とした研究
- 科学技術の全分野で、実験、観測、理論、と並ぶ、重要かつ最先端の研究手段



計算科学の重要性：何に役立つのか

- **「紙と鉛筆」では解けないような複雑な現象の探求**
  - 物質の根源である素粒子の成り立ち
  - DNAやたんぱく質等数百万個の原子の集団の示す性質
- **実験ができない現象の探求**
  - 宇宙における最初の天体の起源
  - 地球規模の気候変動と温暖化予測
- **膨大な大規模データの探索**
  - ゲノムインフォマティクス
- **実験の代替や開発コストの低減**
  - 自動車の衝突シミュレーション
  - 航空機設計
- **AI(機械学習)**
  - 膨大な学習データをスーパーコンピュータで処理する
  - AIを使ってシミュレーションを代替する研究も進んでいる



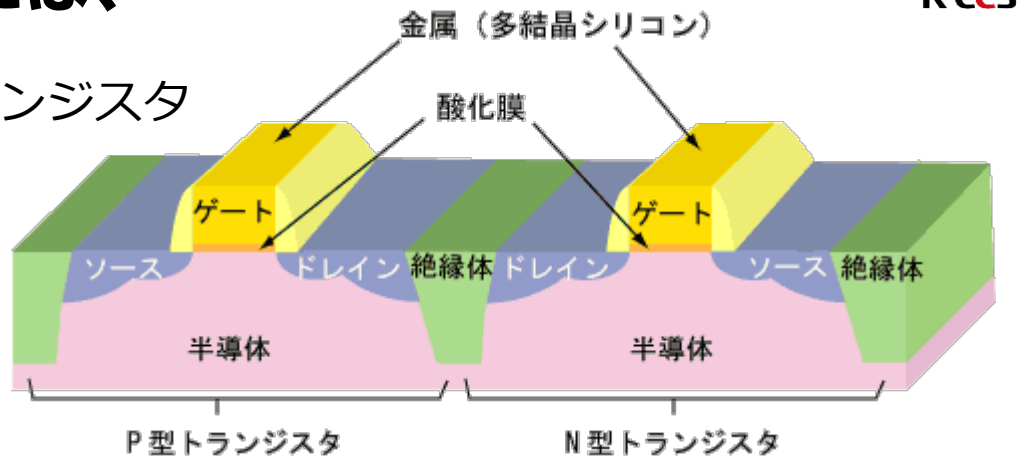
計算機を早くする方法

コンピュータを速くするには、…

①　動作を速くする。

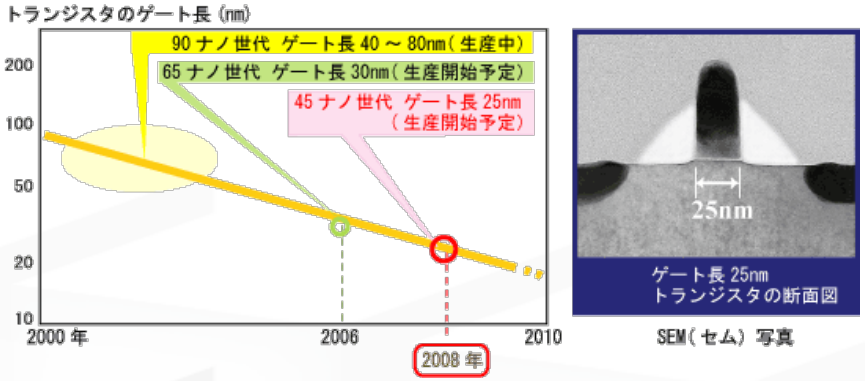
- クロックを速くする
  （PCのプロセッサは2～3GHzの周波数）

CMOSトランジスタ



- 速いトランジスタ（回路）をつかう

小さく作ると、早く動作するが、冷却が問題になる。



https://www.fujitsu.com/jp/group/labs/resources/tech/techguide/list/cmos/p03.html

- マイクロプロセッサ

一つのチップの中にCPUが全部はいったコンピュータ。PCに使われ、今のマイクロプロセッサは、昔のスパコンよりもはやい！

Intel の
プロセッサ

コンピュータを速くするには、…

② コンピュータの中を工夫する

- 一度に、たくさんの命令を実行できるようにする
- メモリの読み出しを早くする… など。

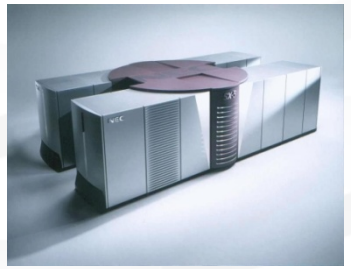
- ベクトル型スーパーコンピュータ

科学技術計算によく使われる行列演算を得意とする計算機
1990年代には、日本のスーパーコンピュータはこのタイプで全世界で広く使われていた。いまのマイクロプロセッサにも同じような機能がある。

Fujitsu VPP500

Fujitsu VPP5000

NEC SX-4



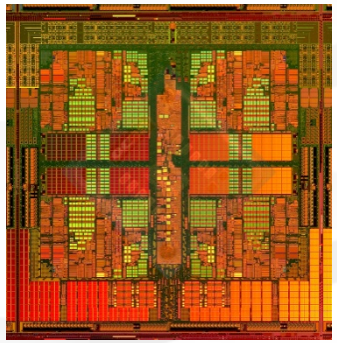
NEC SX-5

コンピュータを速くするには、…

③　たくさんのコンピュータを同時に使う。

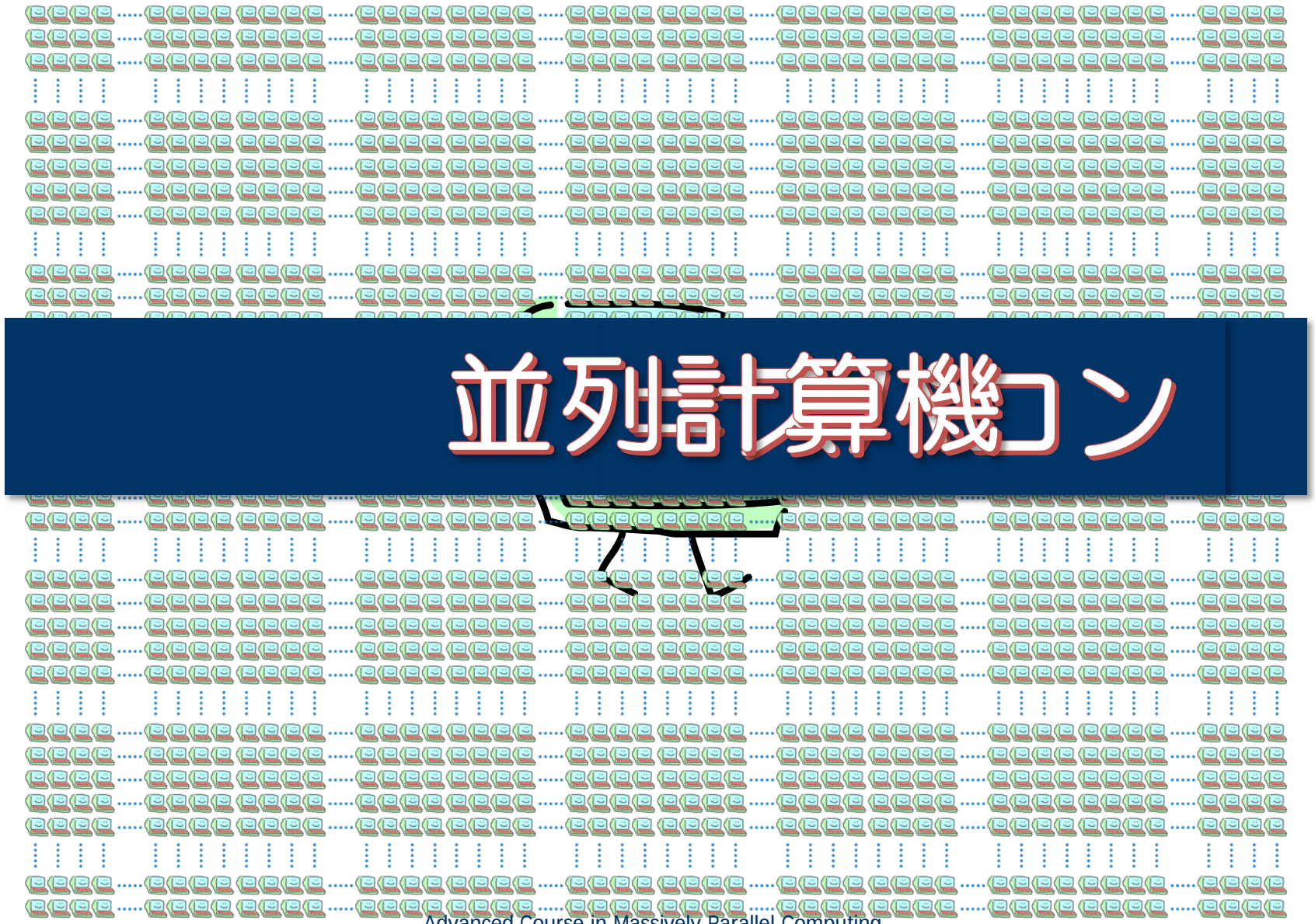
- 並列コンピュータ
- 今のスパコンの主流はこれ！

- PCでも、スマホでも２，　３個のコンピュータがはいっている



AMDのquad coreのプロセッサ
(2008)

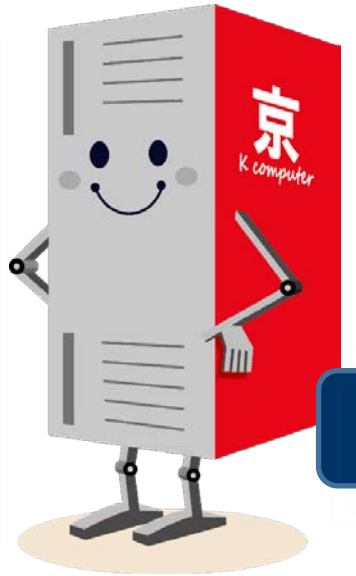
コンピュータ（パソコン）をたくさん繋げている



「京」コンピュータの速さは？

1秒間に10^{16}回の計算ができる
10,000,000,000,000,000 = 1京（1兆の1万倍）

70億人が1秒間に1回計算しても17日



けいたろう
京太郎

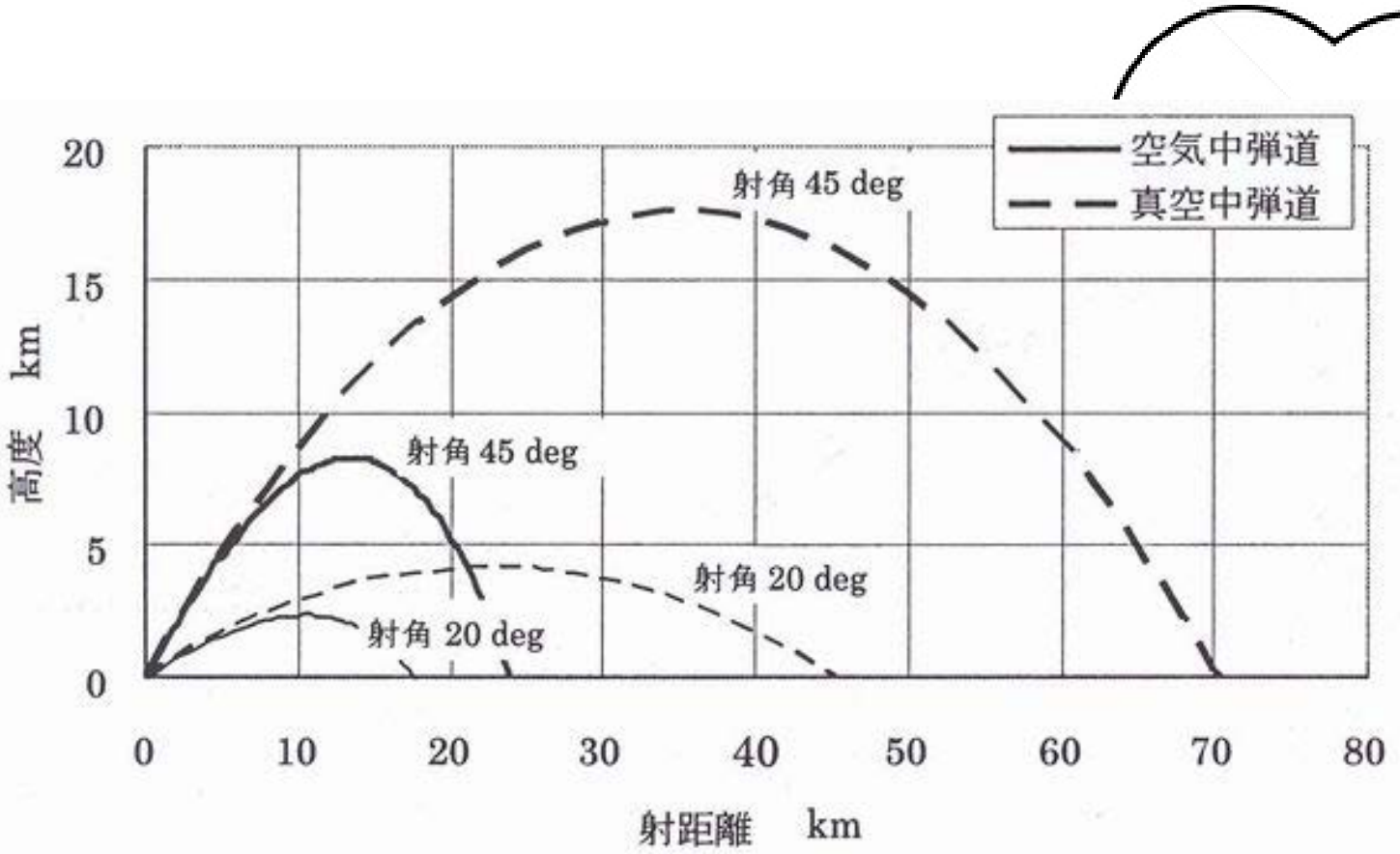
スーパーコンピュータ「富岳」の速さは？

ふがくん

「京」の50～100倍

コンピュータのはじまりは？

- 弾道計算、暗号解読、…



ニュートンの運動方程式

$$F(力) = m(質量) \times a(加速度)$$

$$m(質量) = \frac{F(力)}{a(加速度)}$$

pixta.jp – 2175140

例えば、…

- Fがわかれば、速度V(t) （時刻tの時の速度）は

$$V(t+\Delta t) = V(t) + F(t)\times\Delta t$$

F(t)は時刻tの時の力

- 次々と、Vを計算していけば、同じようにそれぞれの時間でのVが求まる

- 速度が求まれば、位置p(t)も同じように

$$p(t+\Delta t) = p(t) + v(t)\times\Delta t$$

数値予報（天気予報）の場合は…

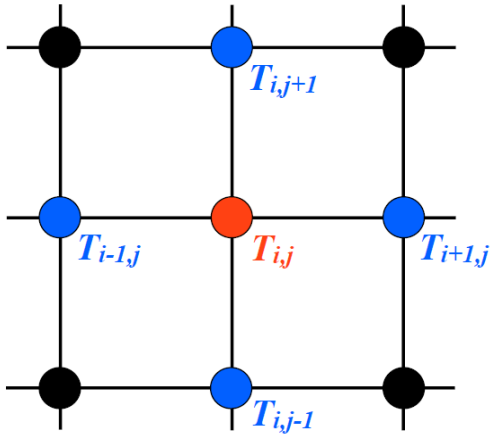
- **天気予報とは、晴れ、曇り、雨などの天気状態と、温度、湿度、風速、気圧などをある領域で、時間経過とともに予測すること。**
  - 晴れ、曇り、雨などの天気状態は、温度や湿度、風速、気圧がわかれば、その状態として考えることができるはず。
  - 温度や湿度、風速、気圧は物理量なので、運動方程式からわかるはず。
  - V. ビァネークス(1904)「力学的、物理学的基礎に立つ問題」

- **空気に働く力＝気圧傾度力、コリオリカ、摩擦力**
  - これをもって、運動方程式をとけばいい…
  - 流体の運動方程式「ナビエ-ストークス方程式」

熱伝導問題の場合

熱拡散の方程式（偏微分方程式）

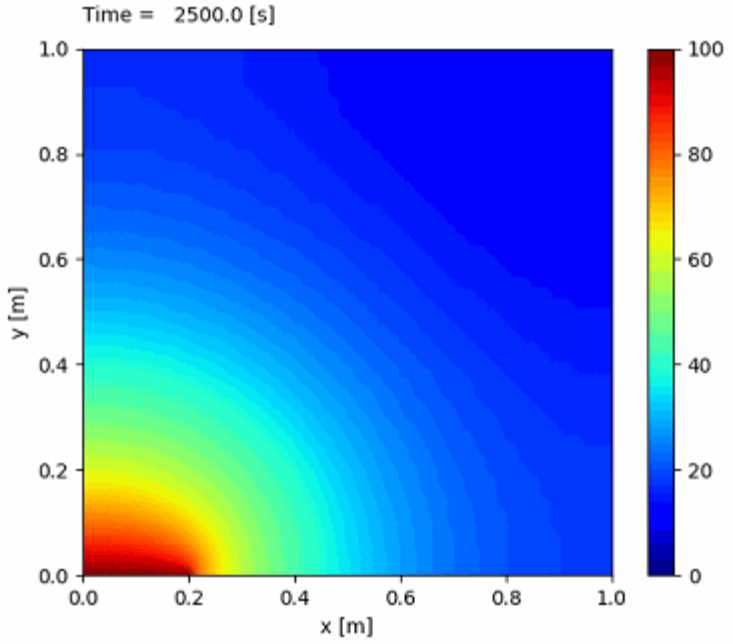
$$\frac{\partial T(x,y,t)}{\partial t} = k\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) T(x,y,t) + s(x,y)$$



「離散化」
格子点上での式にする。

$T(x_i, y_j, t) \Rightarrow T_{i,j}(t)$ と略記。

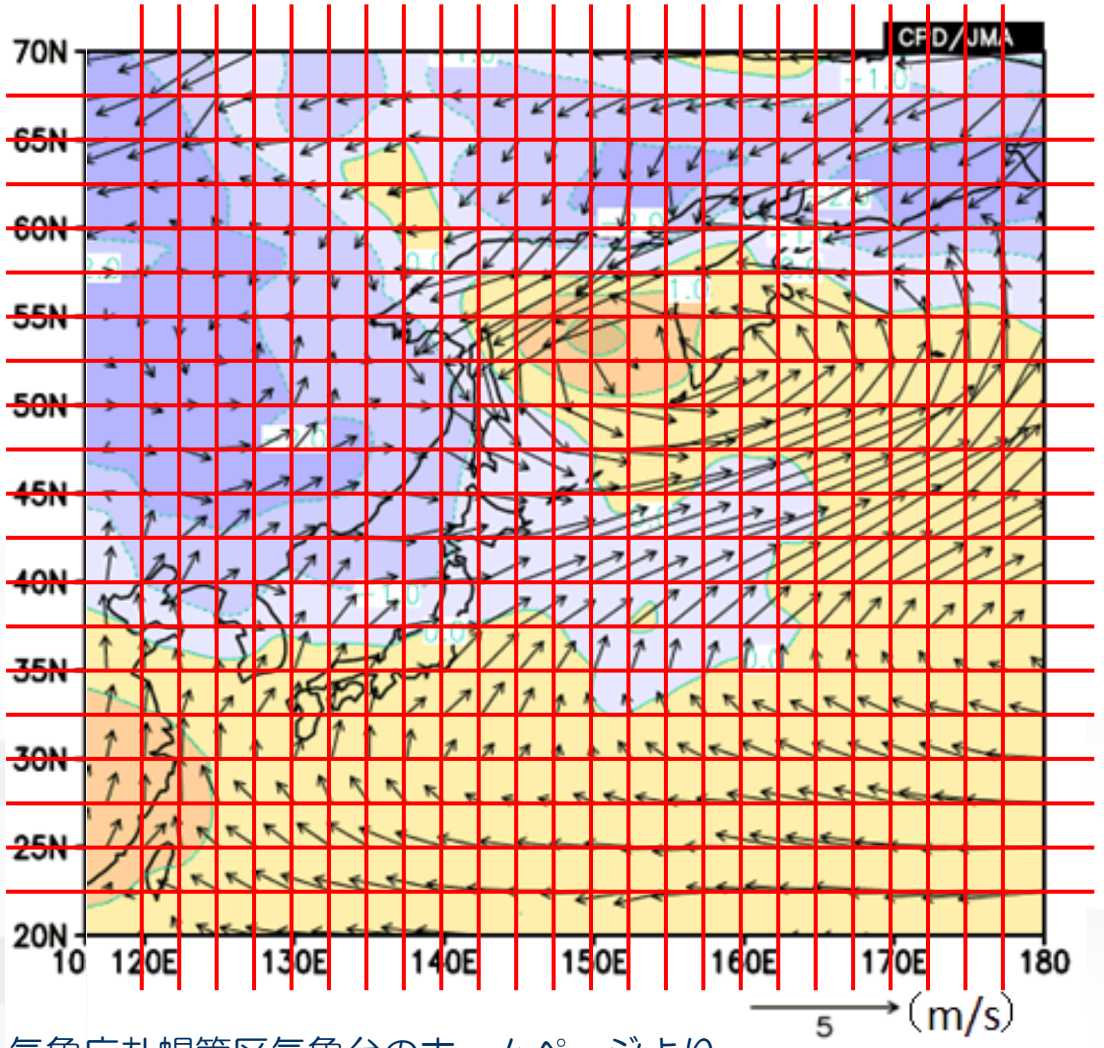
$$\frac{T_{i,j}(t+\Delta t) - T_{i,j}(t)}{\Delta t} = k\left(\frac{T_{i+1,j}(t) - 2\,T_{i,j}(t) + T_{i-1,j}(t)}{\Delta x^2} + \frac{T_{i,j+1}(t) - 2\,T_{i,j}(t) + T_{i,j-1}(t)}{\Delta y^2}\right) + s_{i,j}$$



$$T_{i,j}(t+\Delta t) = T_{i,j}(t) + \frac{k\,\Delta t}{\Delta x^2}\{T_{i+1,j}(t) - 2\,T_{i,j}(t) + T_{i-1,j}(t)\} + \frac{k\,\Delta t}{\Delta y^2}\{T_{i,j+1}(t) - 2\,T_{i,j}(t) + T_{i,j-1}(t)\} + s_{i,j}\,\Delta t$$

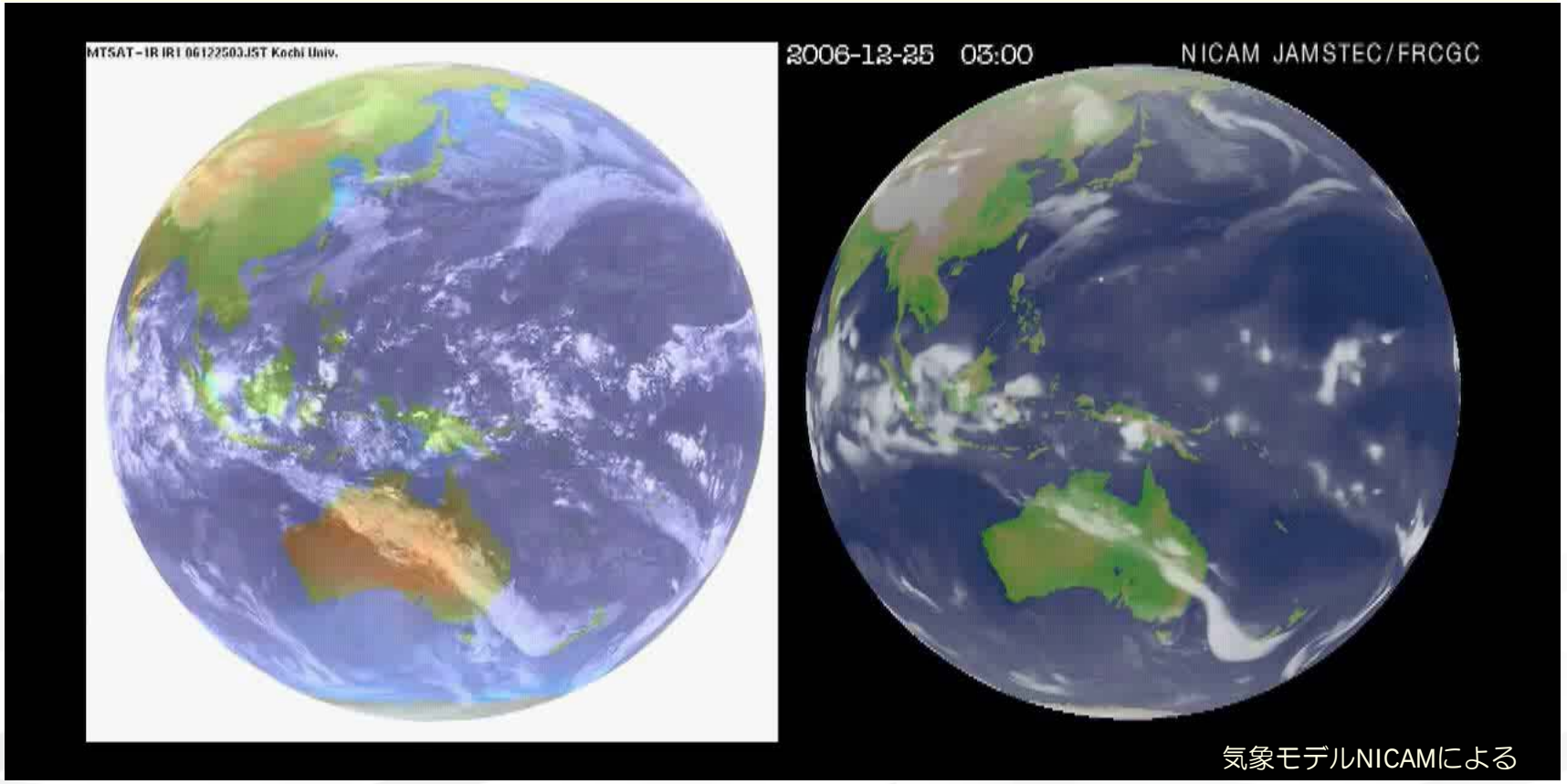
今の時刻tから、 Δt 先 (t+Δ t)のTの値を、四則演算で計算できる。

各メッシュの点で、物理量を計算して天気を予測



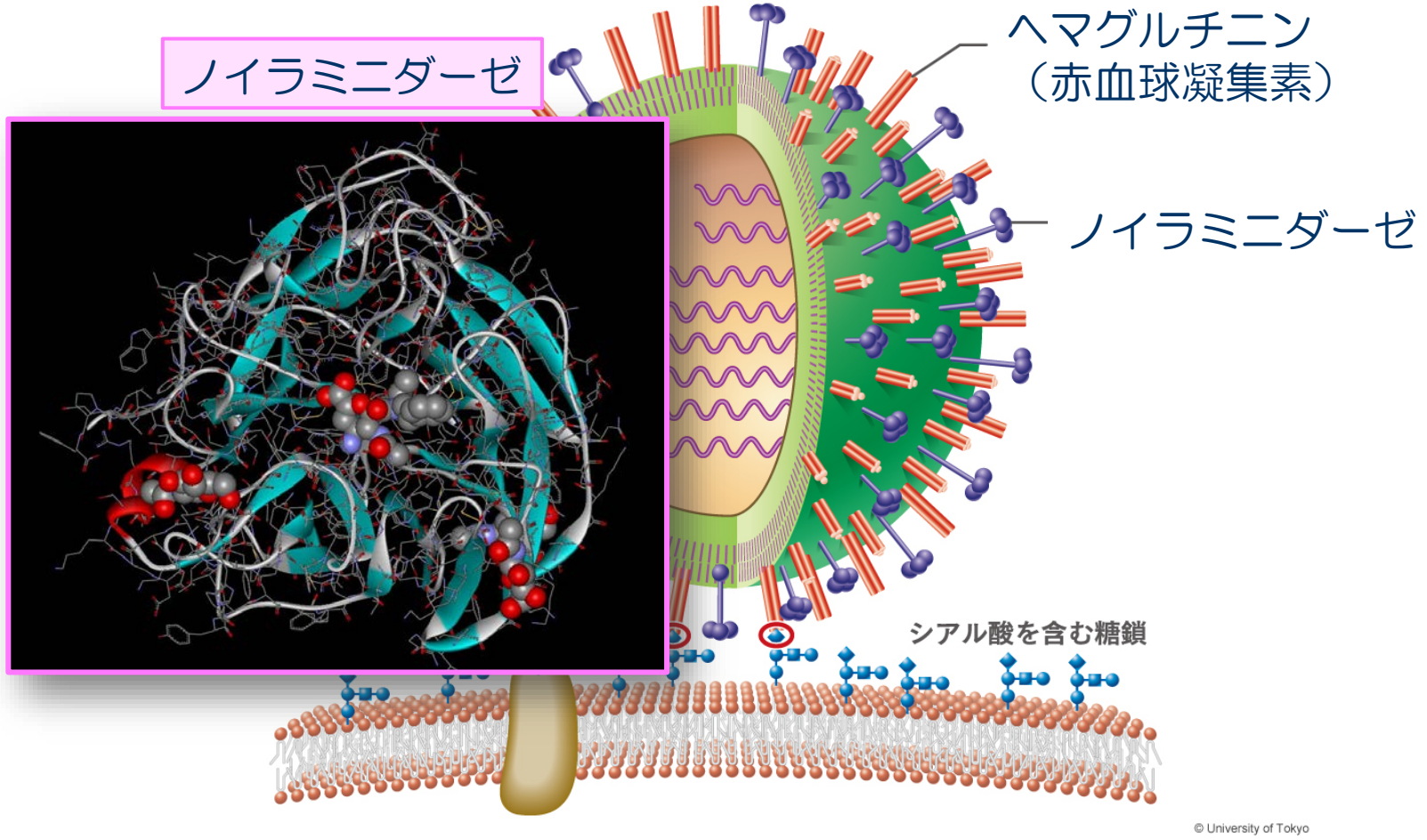
気象庁札幌管区気象台のホームページより
https://www.jma-net.go.jp/sapporo/tenki/kikou/tokucho/kencho2_winter.html

気象衛星 ひまわり6号の画像　*シミュレーションによる画像*



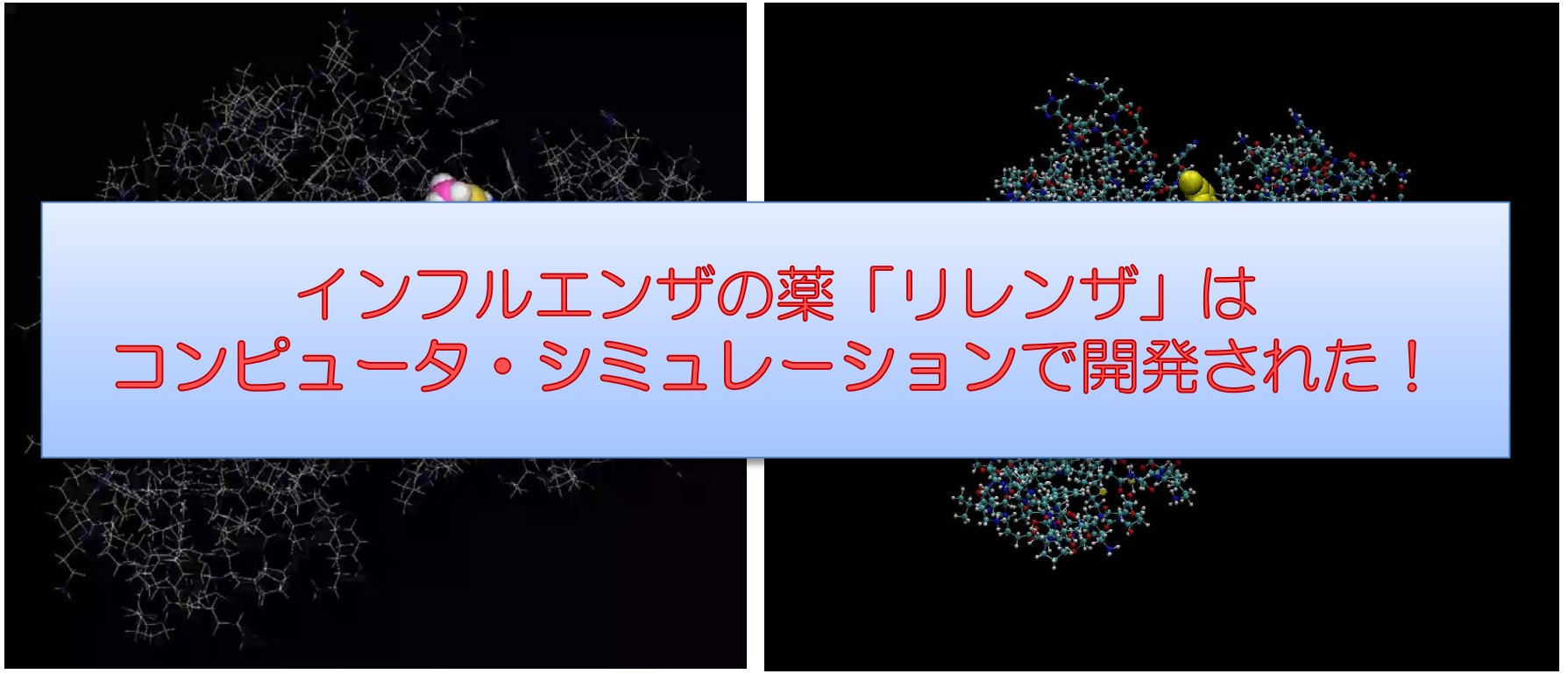
提供：AORI/NIES/JAMSTEC/MEXT

インフルエンザウイルスの働き



出典：生命科学教育用画像集　http://csls-db.c.u-tokyo.ac.jp
インシリコサイエンス社　http://www.pd-fams.com

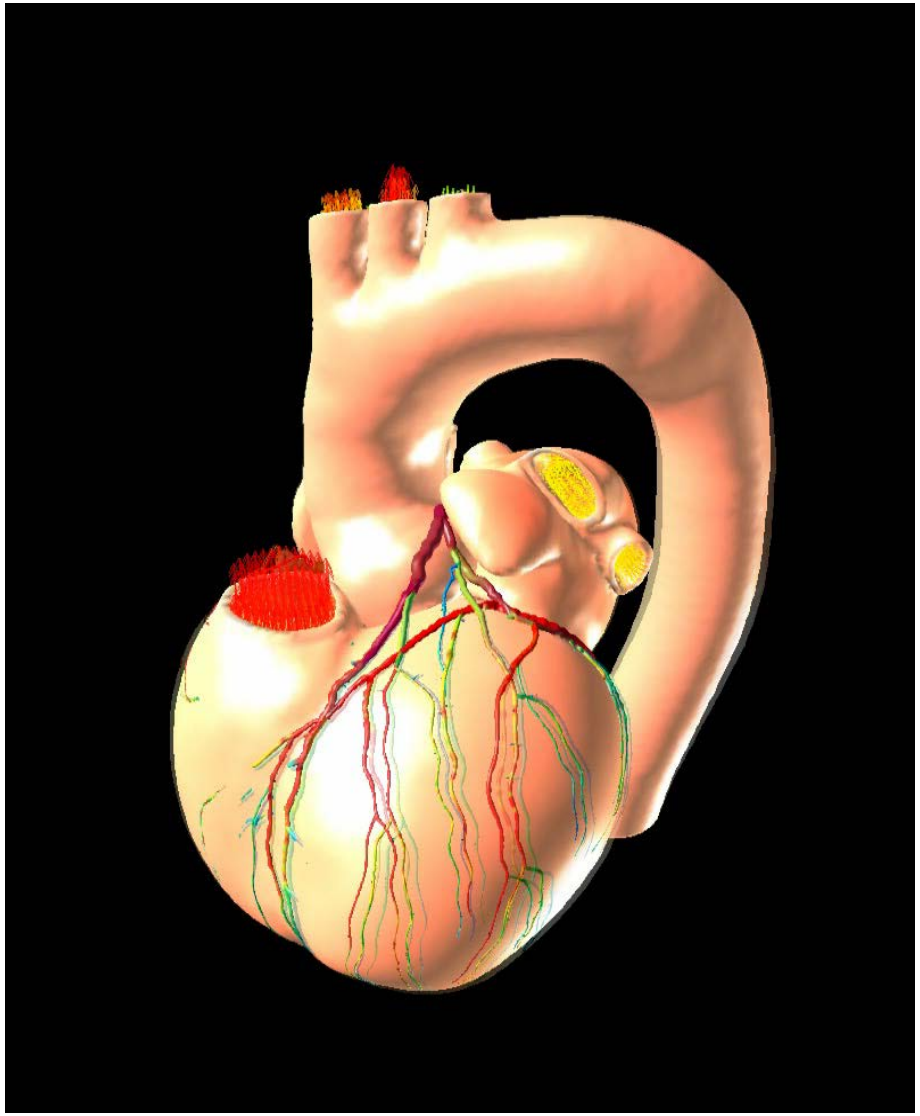
病気の原因となるたんぱく質と薬のドッキングシミュレーション



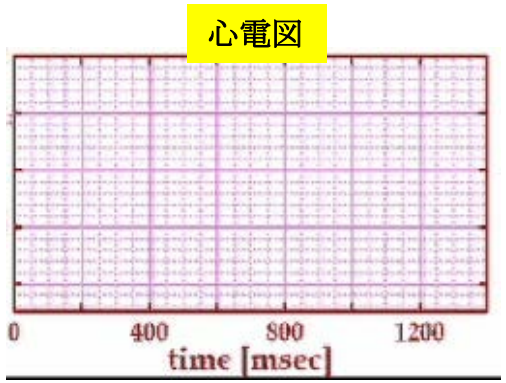
インフルエンザの薬「リレンザ」は
コンピュータ・シミュレーションで開発された！

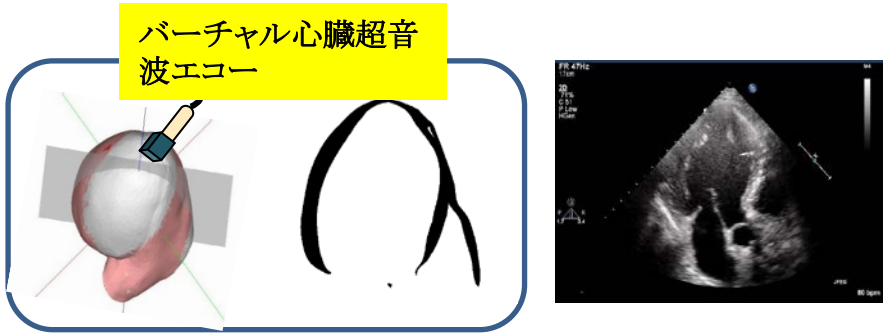
ちょっと前までの計算　　スーパーコンピュータ「京」で計算

細胞モデルからの心臓シミュレーション



株式会社UT-Heart研究所　協力：富士通株式会社

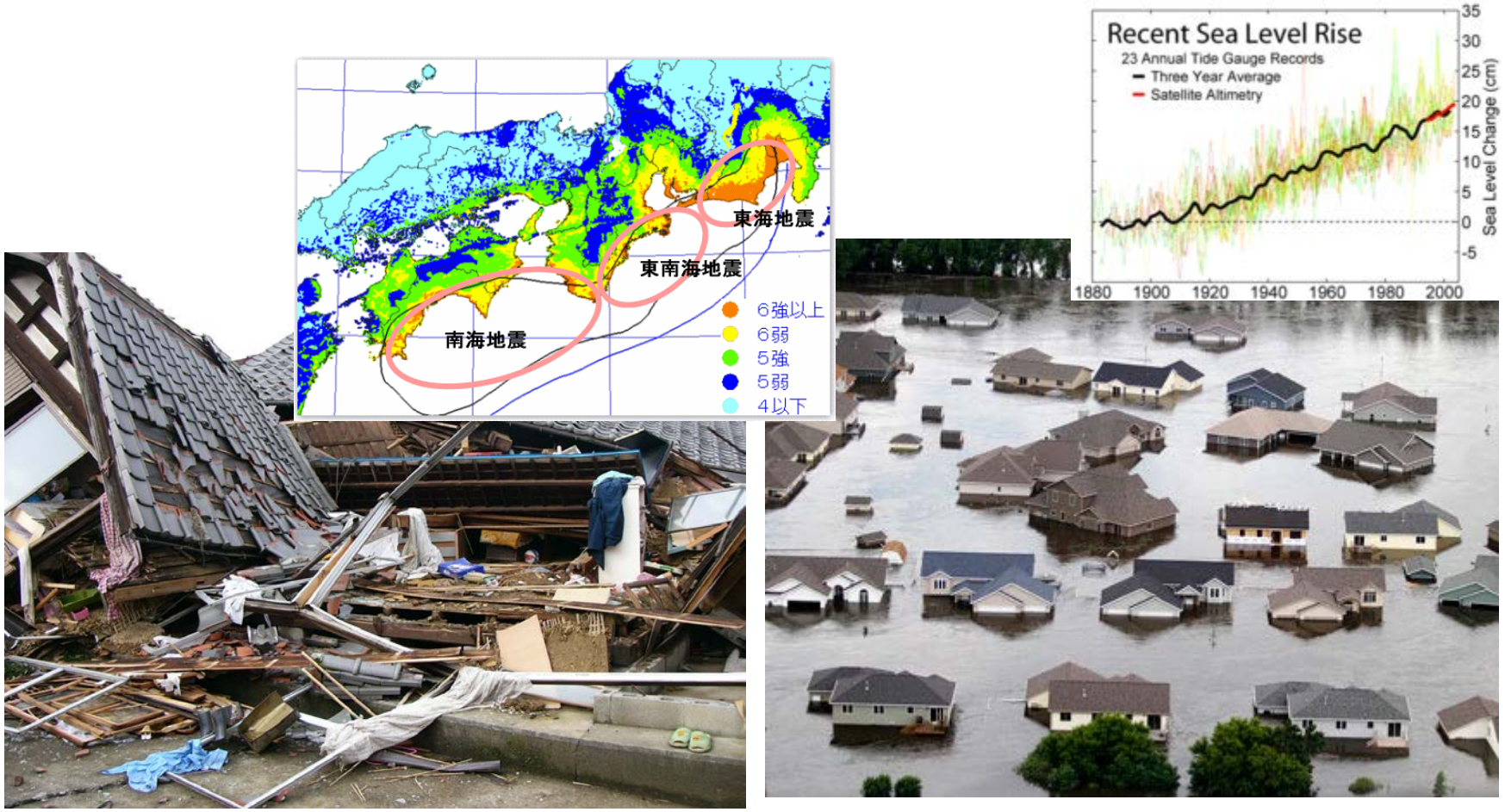




- 心筋細胞内のたんぱく質の確率的運動から細胞の収縮、心拍動、血液駆出、冠循環までを一貫してシミュレート。
- シミュレーションから超音波エコー、流速ドップラー、心電図、カテーテル検査などの精緻なデータが再現される。そのデータを基に病態の解析が可能に。仮想手術や薬の副作用予測（不整脈予測）にも応用。

スパコンは役にたつのか…

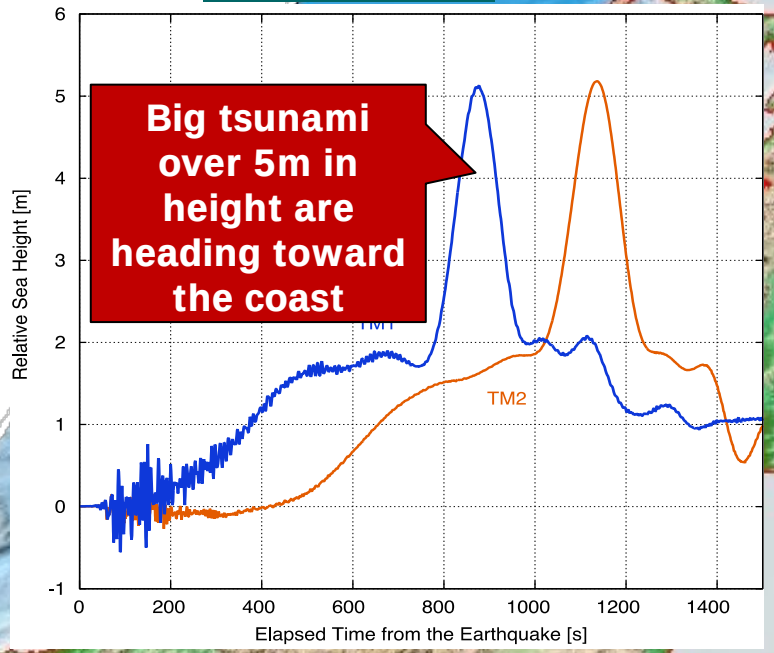
- これからの科学技術の発展に不可欠
- 防災や温暖化問題にも、…



The tsunamis were captured by seafloor pressure gauge

Courtesy: T. Furumura (U. Tokyo)

Sea height data

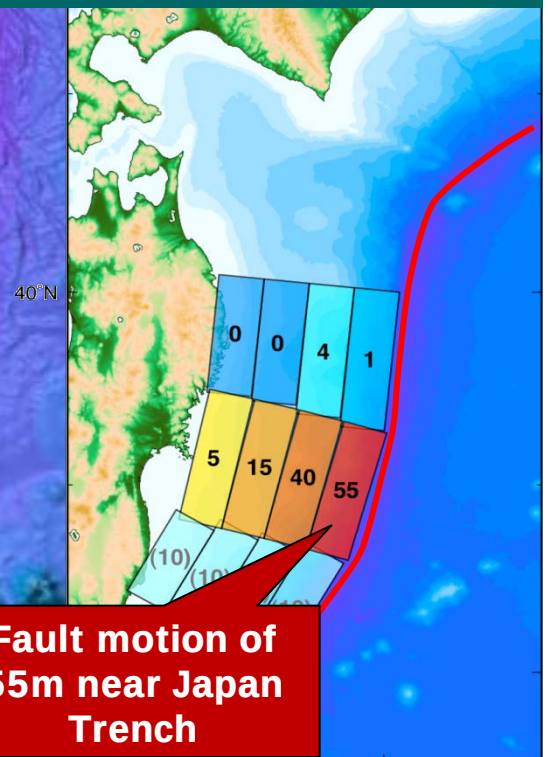


Big tsunami over 5m in height are heading toward the coast

Seafloor cable tsunami gauge off Kamaishi City at 50 and 80km

Estimated fault slip motion

Maeda et al. (2011)



Tsunami analysis

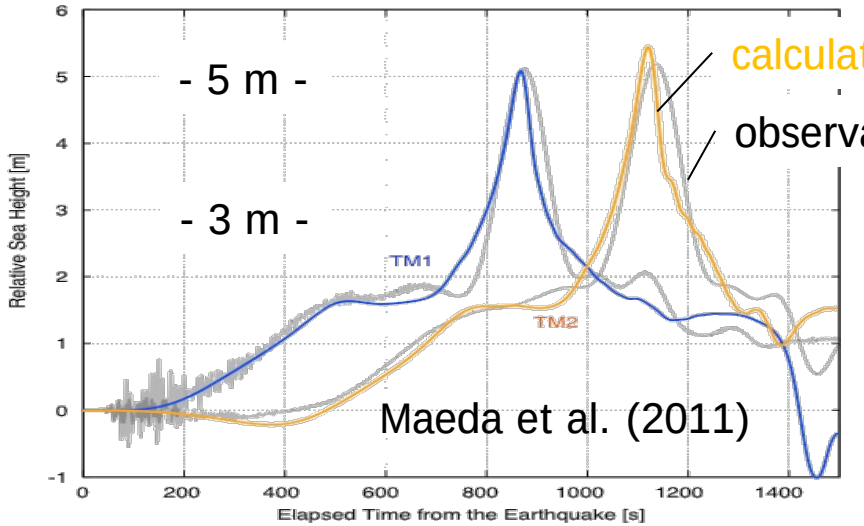
Fault motion of 55m near Japan Trench

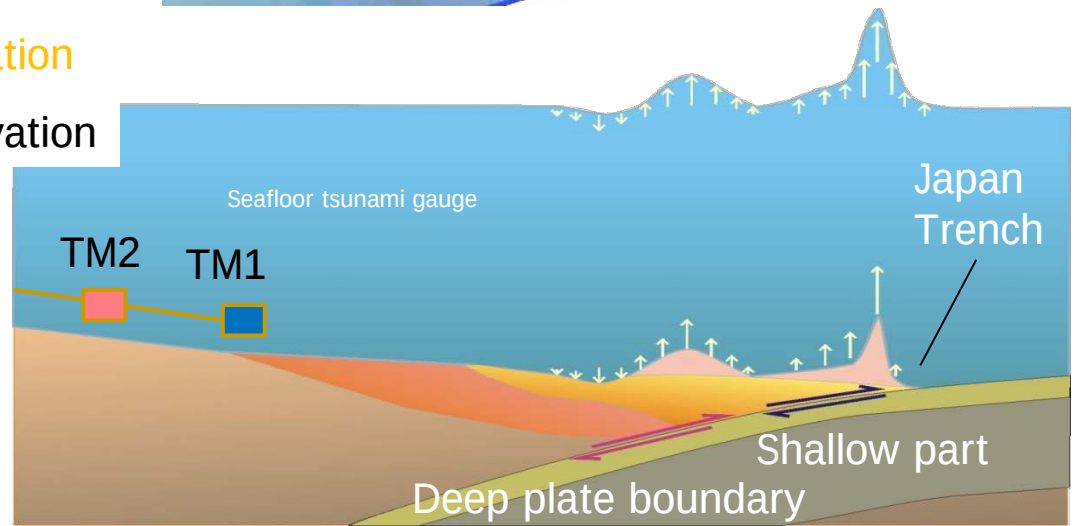
The combination of deep and shallow plate slips generated the big tsunamis

Courtesy:
T. Furumura (U. Tokyo)

(a) Slip of deep plate boundary only

(b) Slip of deep *and* shallow plate boundary





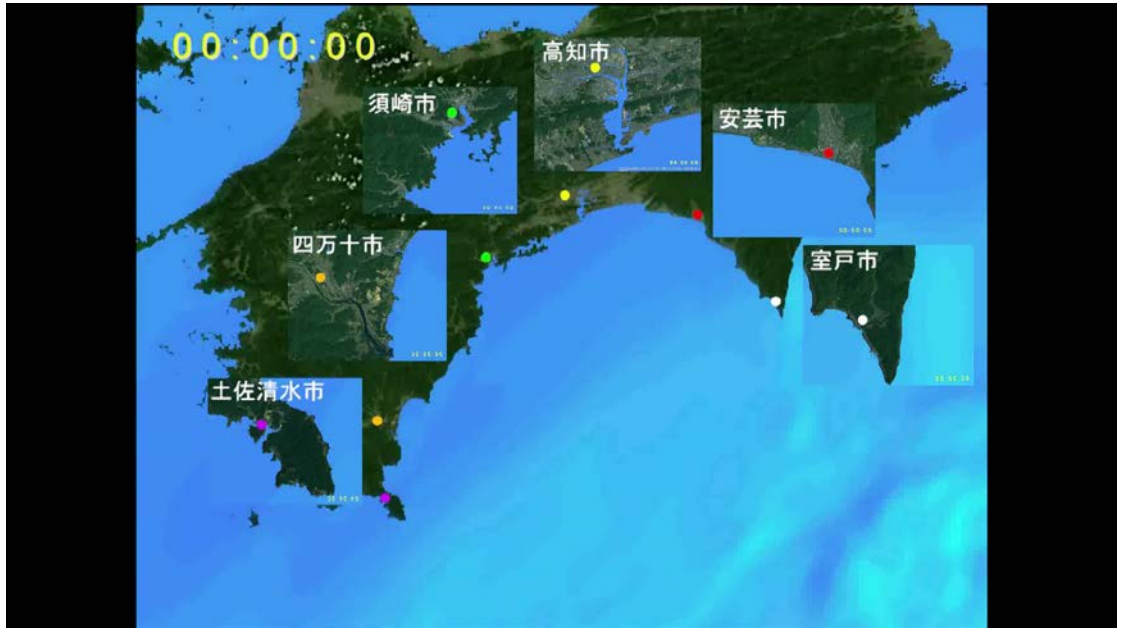
高精細なシミュレーションによる災害に対する防災・減災

巨大地震により引き起こされる①強い揺れ，②地殻変動（海底や海岸の隆起・沈降），そして③津波を，地震発生からの時間を追って詳細に評価して地震防災・避難計画に活用するために「地震 - 津波同時シミュレーション」を開発

東日本大震災の再現

前田、古村（東大）

南海トラフ地震・津波の予測



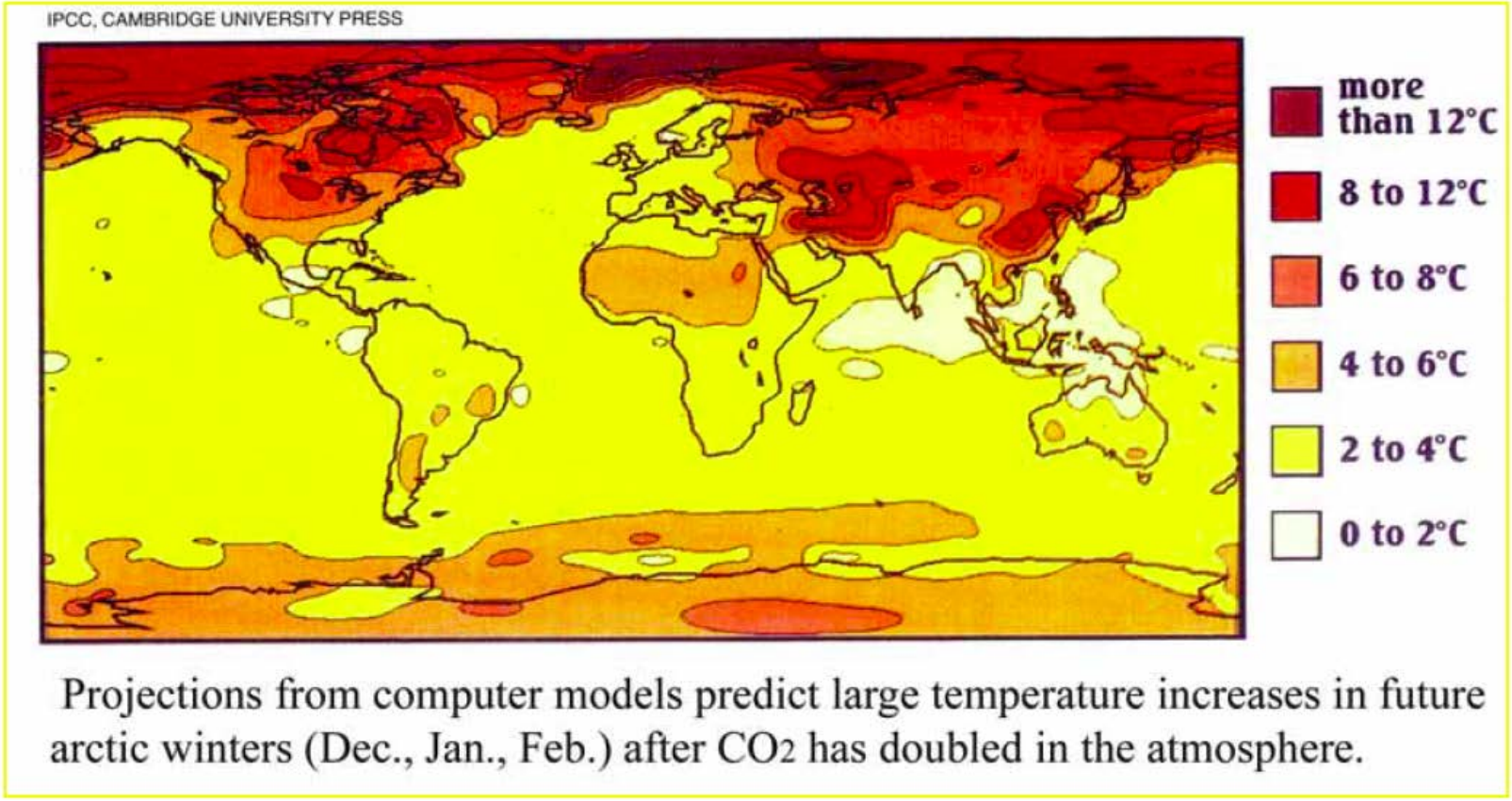
南海トラフ巨大地震　広域詳細な津波計算　　馬場 (JAMSEC)

たくさんのシナリオを用意し、地方自治体と連携し、防災計画、ハザードマップの作成に寄与

「地球シミュレータ」の結果

CO2倍増実験

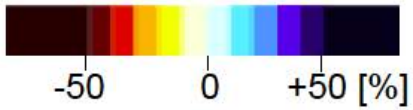
大気中の二酸化炭素増加を想定したコンピューターの気候予測は
特に冬期の北極圏の気温上昇を予測している。



Projections from computer models predict large temperature increases in future arctic winters (Dec., Jan., Feb.) after CO_2 has doubled in the atmosphere.

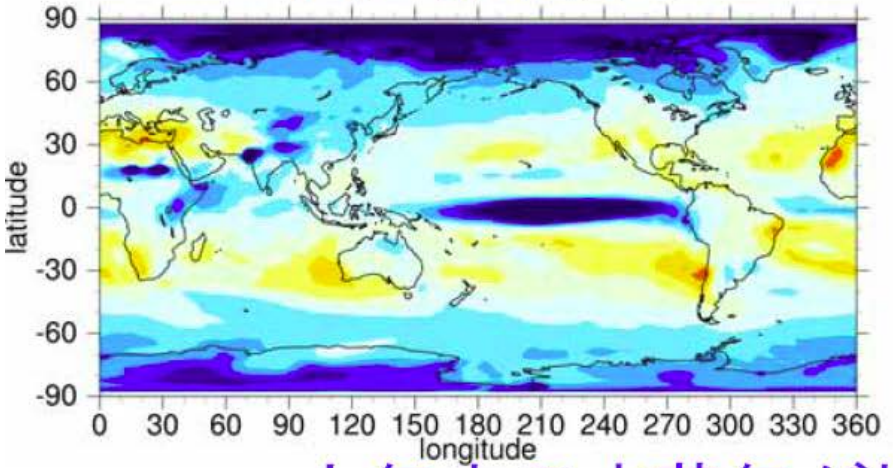
「地球シミュレータ」の結果

温暖化による平均降水量および豪雨強度の変化率

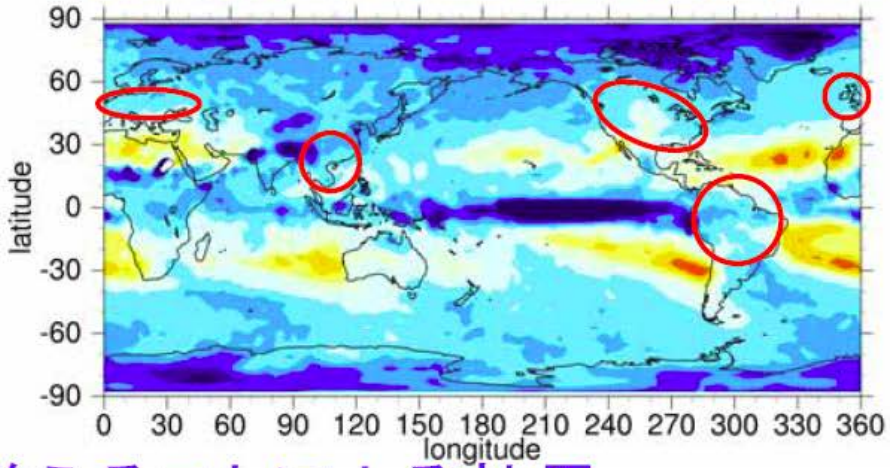


赤丸をつけた領域は、平均降水量は増えないが、豪雨強度は増える
これは水蒸気の増加による効果が場所によって異なるため

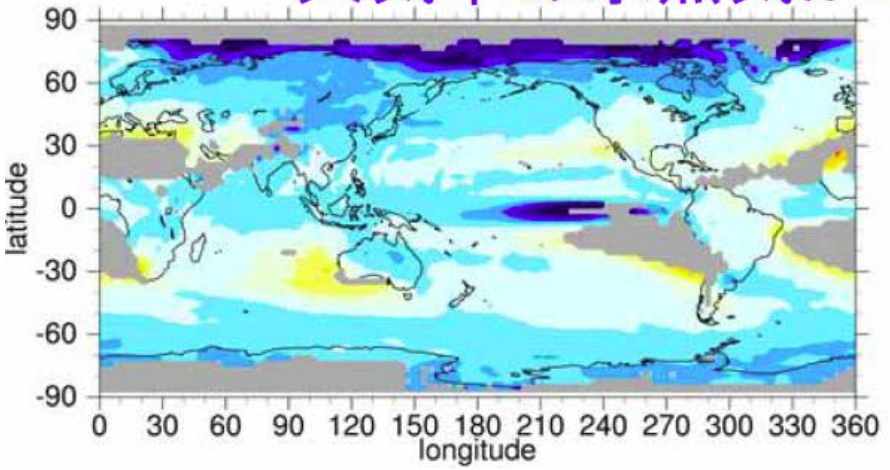
年平均降水量

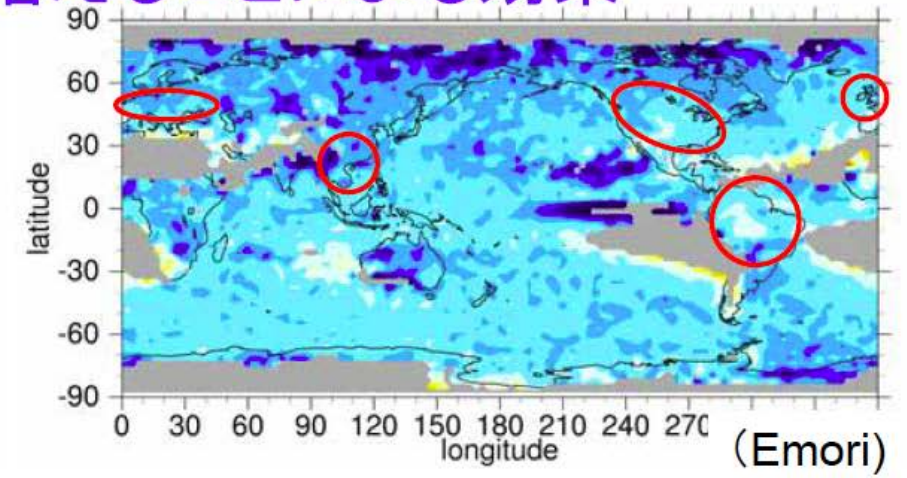


豪雨強度（99%ile日降水量）



大気中の水蒸気が増えることによる効果



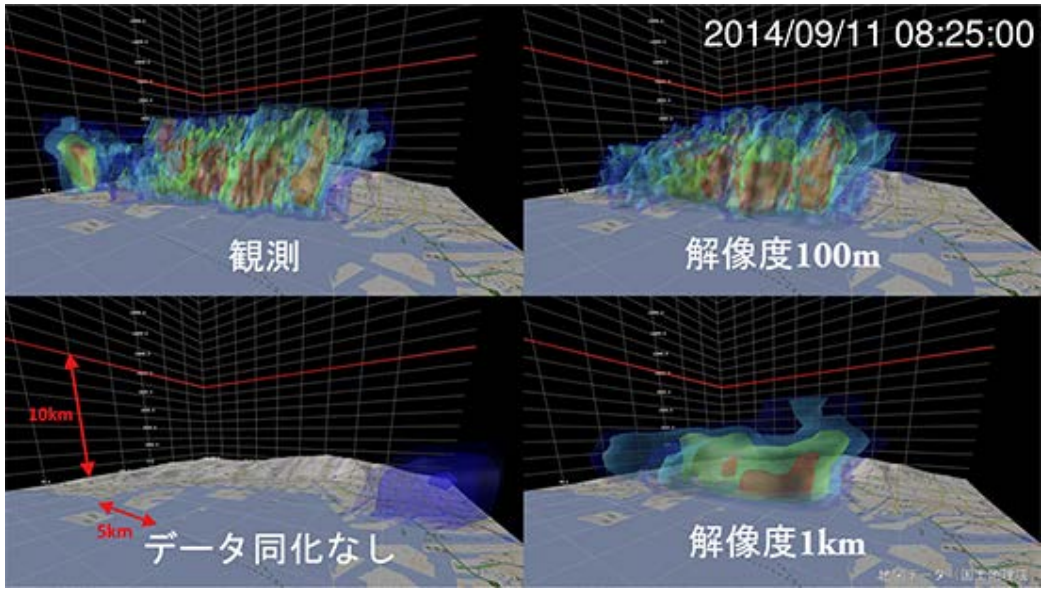


（Emori）

「京」と最新鋭気象レーダを組み合わせたゲリラ豪雨予測

- 現在の天気予報は、2kmの解像度でシミュレーションを行い、1時間毎に新しい観測データを取り込んで更新するため、わずか数分の間に局地的にゲリラ豪雨を引き起こす積乱雲を予測することは困難。

- 「京」を使った解像度100mの高精細シミュレーションに30秒毎の観測データを組み合わせた時間的・空間的に桁違いのシミュレーションを世界で初めて実現し、実際のゲリラ豪雨の動きを詳細に再現することに成功。



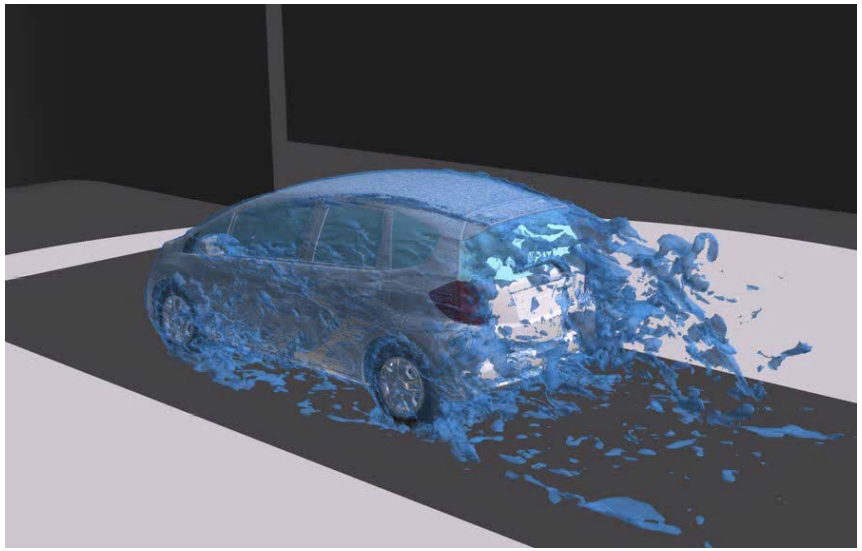
〈2014年9月11日午前8時25分の神戸市付近における雨雲の分布〉

解像度100mのシミュレーション結果は、積乱雲内部の微細構造や降水分布が観測データに非常に近いことが分かる。

「京」が拓いたHPC-CAE（風洞実験主体からスパコン活用設計へ）

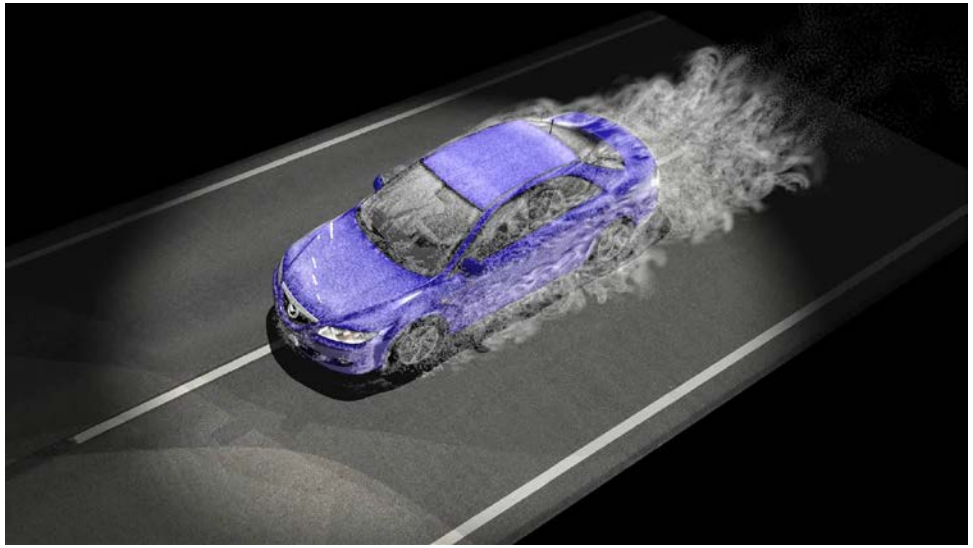
- 風洞実験に匹敵するシミュレーションの実現（対実験誤差1～2%）（左）
- 風洞ではできない、走行中に遭遇するリスク・安定性シミュレーション（追い越し、追い抜き、突風、急ハンドル操作）の実現（右）
- 世界中の自動車会社等（例：シュツットガルト大学自動車研究所、ベンツ社）が同様のシミュレーションに取り組んでいるが、「京」の1/10以下の規模の解析しかなされていない。

※2017年自動車用次世代 CAE（Computer Aided Engineering）コンソーシアムを設立、13法人7機関が参画。



神戸大学、理研、ホンダ提供

風洞代替シミュレーションの国内経済効果
風洞実験主体の開発からシミュレーション主体の計測へ。これにより開発費用の削減や空気抵抗低減が加速。



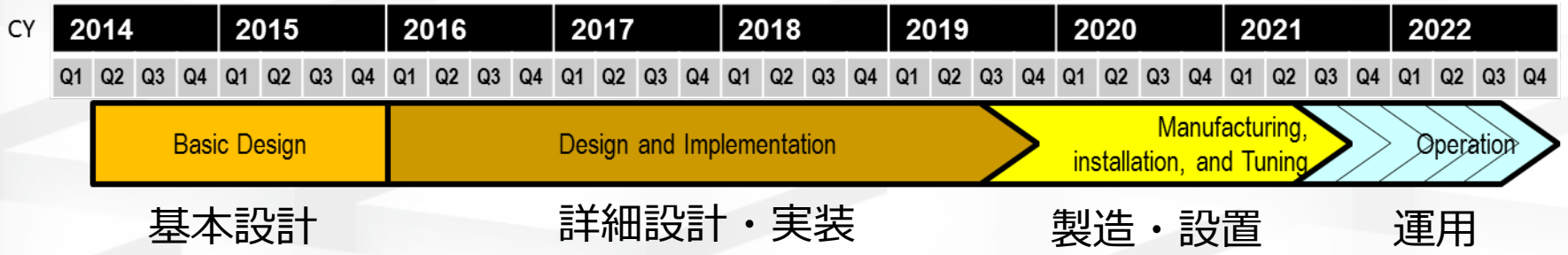
神戸大学、広島大学、理研、マツダ提供

リアルワールドシミュレーションの国内経済効果
開発最終段階で評価していたリアルワールド評価を上流段階で、これにより設計変更を減らすことが可能。

「京」から「富岳」へ

フラッグシップ2020プロジェクト

- **目的**
  - 「京」の次のスーパーコンピュータ（ポスト「京」＝「富岳」）の開発
  - ポスト京で我が国の科学および社会的・諸課題を解決するアプリケーションの開発
- **プロジェクトは、2014年度から開始**
  - 理研が開発主体
  - ベンダーパートナーは、富士通
- **2020年度末で開発プロジェクトは終了、2021年３月から共用開始**



FLAGSHIP 2020 プロジェクトにおける開発目標

Post-K(富岳）の開発については、次の3つの開発目標を設定し、 設計を進めた。

- **1. 高い電力効率**
  - 施設の最大電力量は、30 - 40MW (for system)に設定し、電力設備を準備した
- **2.「ターゲット・アプリケーション」での高効率・高性能**
  - 目安として、いくつかのアプリケーションで、「京」の100倍以上の性能を達成できること。
  - ベンチーマーク（Top500等）は目標ではない。
- **3. 広いユーザーに対して、使いやすいシステム**
  - ＧＰＵなどのアクセラレータは不採用
  - 京でのアプリケーションが継続して使える

Codesign of “Fugaku”

3 Design Targets:

- **1. Extreme Power-Efficient System**
  - Maximum performance under Power consumption of 30 - 40MW (for system)
- **2. Effective performance of target applications**
  - It is expected to exceed 100 times higher than the K computer’s performance in some applications
- **3. Ease-of-use system for wide-range of users**

Cool (Low-power) technology is important!!

Codesign

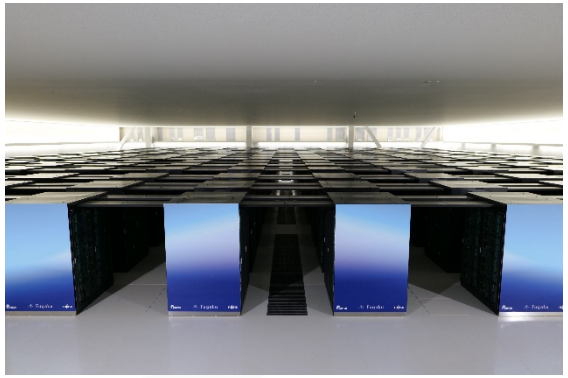
Codesign to meet these 3 design targets

Technologies and Architectural Parameters to be determined

- **Basic Architecture Design (by Feasibility Studies)**
  - Manycore approach, O3 cores, some parameters on chip configuration and SIMD
- **Instruction Set Architecture and SIMD Instructions**
  - Fujitsu collaborated with Arm, contributing to the design of the SVE as a lead partner
- **Chip configuration**
- **Memory technology**
  - DDR, HBM, HMC …
- **Cache structure**
- **Out of order (O3) resources**
- **Enhancement for Target Applications**
- **Interconnect between Nodes**
  - SerDes, topologies “Tofu” or other network?

- ✓ The number of cores in a CMG
- ✓ The number of CMGs in a chip
- ✓ How to connect cores to shared L2 in a CMG
- ✓ The number of ways, the size, and throughputs of the L1
- ✓ and L2 caches
- ✓ The topology of network-on-chip to connect CMGs
- ✓ The die size of the chip
- ✓ The number of chips in a node

Overview of “Fugaku” and A64FX processor



Supercomputer “Fugaku” and A64FX processor

- **Ultra-scale “general-purpose” manycore system: 158,976 nodes (1 processor/node, total 7.6 M cores, theoretical peek 537PFLOPS (DP))**
- **Arm-based manycore processor: Fujitsu A64FX (Armv8.2-A SVE 512bit SIMD, #core 48 + 2/4, 3TF@2.0GHz, boost to 2.2GHz)**
  - 12 cores in a cluster of cores called CMG, connected to L2 and HBM memory chips
- **Advanced Memory technology: HBM2 32 GiB, 1024 GB/s bandwidth, packaged in CPU chip**
- **Scalable Interconnect: ToFu-D interconnect**

- Standard programing model is OpenMP-MPI hybrid programming. running each MPI process on a NUMA node (CMG).
- 48 threads OpenMP is also supported.

CMG(Core-Memory-Group): NUMA node
12+1 core

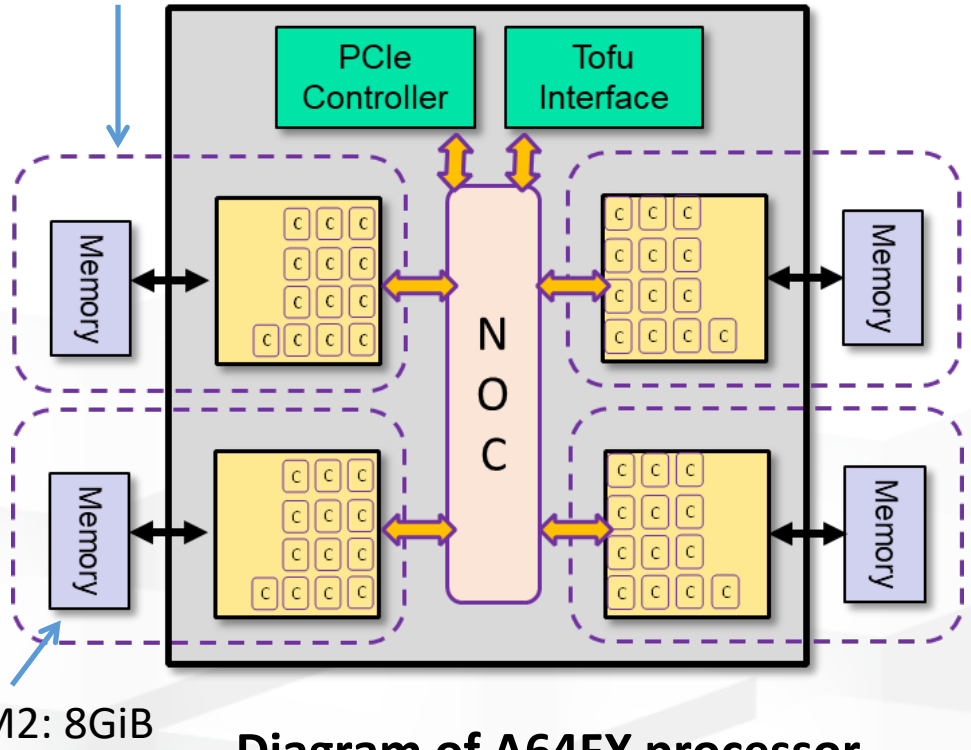
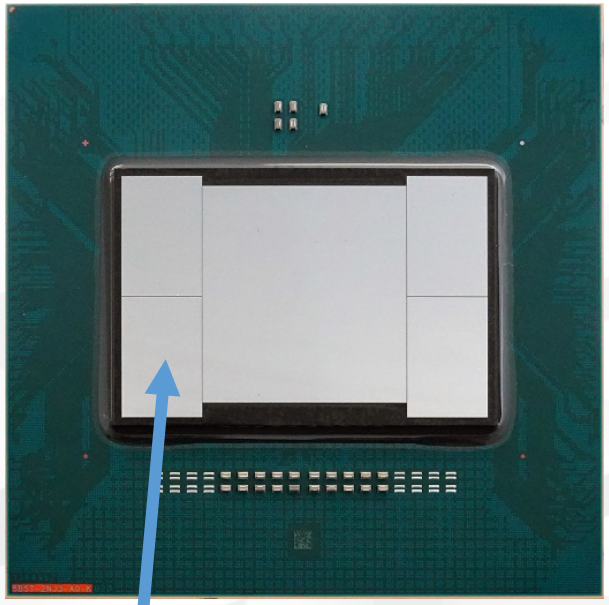


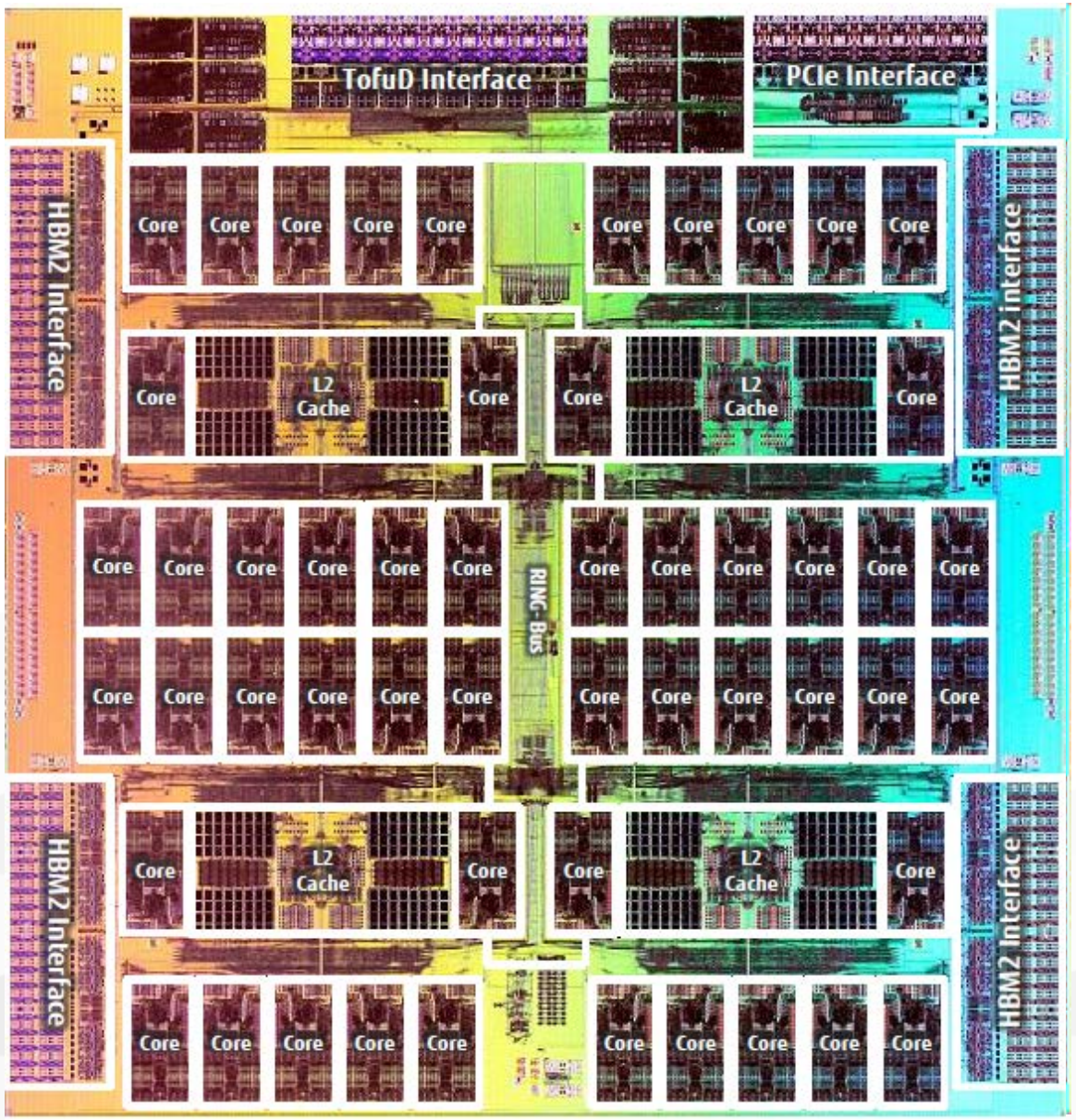
Diagram of A64FX processor

Die Photograph of A64FX processor

- TSMC 7nm FinFET
- 400 mm^2
- HBM2 chips are mounted on Si-interposer connected by TSMC CoWoS technology



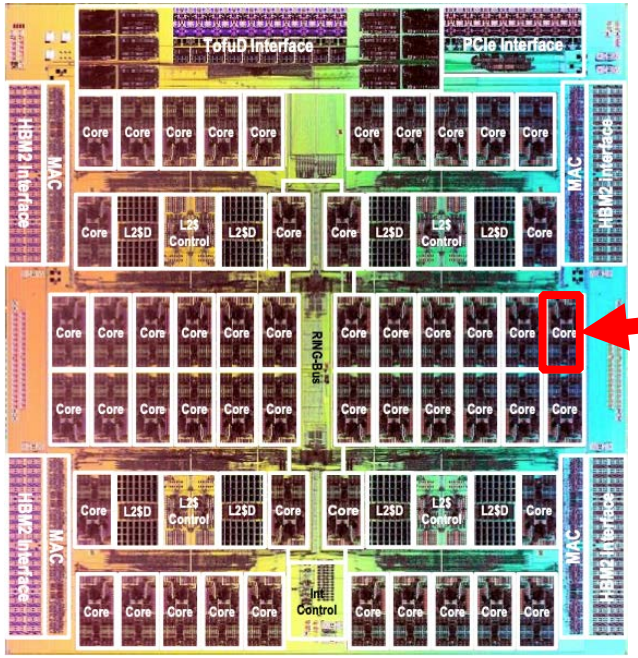
HBM2



Comparison of Die-size

- **A64FX: 52 cores (48 cores), 400 mm² die size (8.3 mm²/core), 7nm FinFET process (TSMC)**
- **Xeon Skylake: 20 tiles (5x4), 18 cores, ~485 mm² die size (estimated) (26.9 mm²/core), 14 nm process (Intel)**
- **A64FX core is more than 3 times smaller per core.**

A64FX:
400 mm²
(20 x 20)



https://www.fujitsu.com/jp/solutions/business-technology/tc/catalog/ff2019-post-k-computer-development.pdf

https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)

Xeon Skylake, High Core Count:
4 x 5 tiles, 18 cores, 2 tiles used for memory interface
485 mm² (22 x 22)

ARM v8 Scalable Vector Extension (SVE)

- **SVE is a complementary extension that does not replace NEON, and was developed specifically for vectorization of HPC scientific workloads.**
- **The new features and the benefits of SVE comparing to NEON**
  - **Scalable vector length (VL)** : Increased parallelism while allowing implementation choice of VL
  - **VL agnostic (VLA) programming**: Supports a programming paradigm of write-once, run-anywhere scalable vector code
  - **Gather-load & Scatter-store**: Enables vectorization of complex data structures with non-linear access patterns
  - **Per-lane predication**: Enables vectorization of complex, nested control code containing side effects and avoidance of loop heads and tails (particularly for VLA)
  - Predicate-driven loop control and management: Reduces vectorization overhead relative to scalar code
  - Vector partitioning and SW managed speculation: Permits vectorization of uncounted loops with data-dependent exits
  - Extended integer and floating-point horizontal reductions: Allows vectorization of more types of reducible loop-carried dependencies
  - Scalarized intra-vector sub-loops: Supports vectorization of loops containing complex loop-carried dependencies

SVE example

DAXPY (scalar)

```
// ---------------------------------------
//       subroutine daxpy(x,y,a,n)
//       real*8 x(n),y(n),a
//       do i = 1,n
//          y(i) = a*x(i) + y(i)
//       enddo
// ---------------------------------------
// x0 = &x[0], x1 = &y[0], x2 = &a, x3 = &n
daxpy_:
    ldrsw    x3, [x3]              // x3=*n
    mov      x4, #0                // x4=i=0
    ldr      d0, [x2]              // d0=*a
    b        .latch
.loop:
    ldr      d1, [x0,x4,lsl 3]     // d1=x[i]
    ldr      d2, [x1,x4,lsl 3]     // d2=y[i]
    fmadd    d2, d1, d0, d2        // d2+=x[i]*a
    str      d2, [x1,x4,lsl 3]     // y[i]=d2
    add      x4, x4, #1            // i+=1
.latch:
    cmp      x4, x3                // i < n
    b.lt     .loop                 // more to do?
    ret
```

DAXPY (SVE)

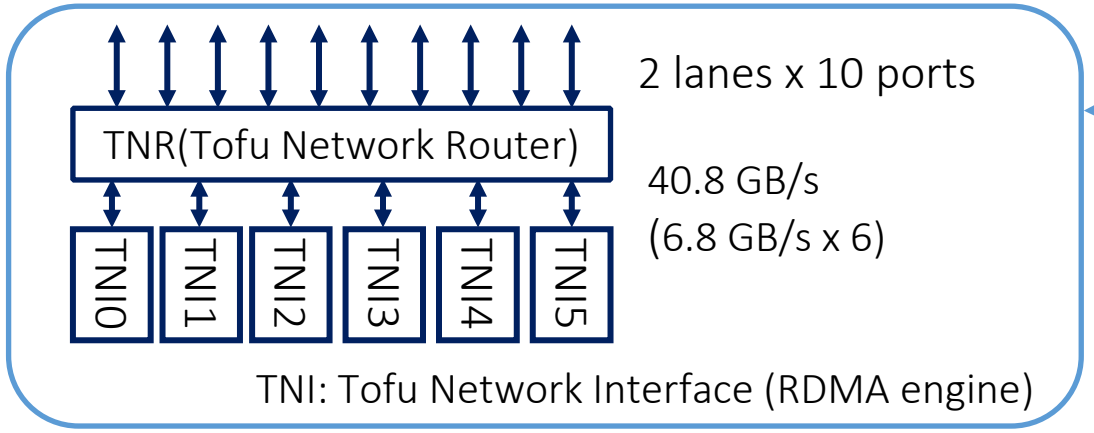
```
// ---------------------------------------
//       subroutine daxpy(x,y,a,n)
//       real*8 x(n),y(n),a
//       do i = 1,n
//          y(i) = a*x(i) + y(i)
//       enddo
// ---------------------------------------
// x0 = &x[0], x1 = &y[0], x2 = &a, x3 = &n
daxpy_:
    ldrsw    x3, [x3]              // x3=*n
    mov      x4, #0                // x4=i=0
    whilelt  p0.d, x4, x3          // p0=while(i++<n)
    ld1rd    z0.d, p0/z, [x2]      // p0:z0=bcast(*a)
.loop:
    ld1d     z1.d, p0/z, [x0,x4,lsl 3] // p0:z1=x[i]
    ld1d     z2.d, p0/z, [x1,x4,lsl 3] // p0:z2=y[i]
    fmla     z2.d, p0/m, z1.d, z0.d    // p0?z2+=x[i]*a
    st1d     z2.d, p0, [x1,x4,lsl 3]   // p0?y[i]=z2
    incd     x4                        // i+=(VL/64)
.latch:
    whilelt  p0.d, x4, x3          // p0=while(i++<n)
    b.first  .loop                 // more to do?
    ret
```

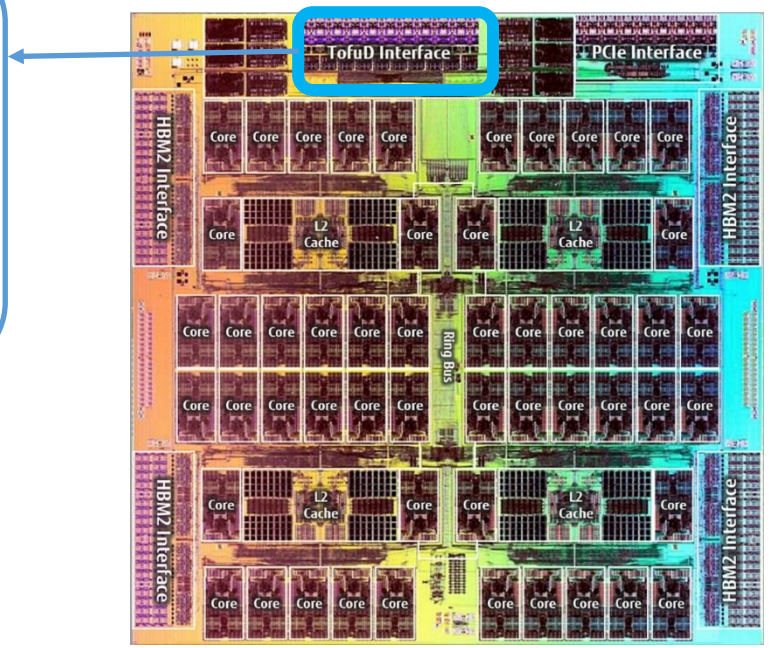
Make predicate mask

SIMD with mask

- **Compact code for SVE as scalar loop**
- **OpenMP SIMD directive is expected to help the SVE programming**

TofuD Interconnect





- 6 RDMA Engines
- Hardware barrier support
- Network operation offloading capability

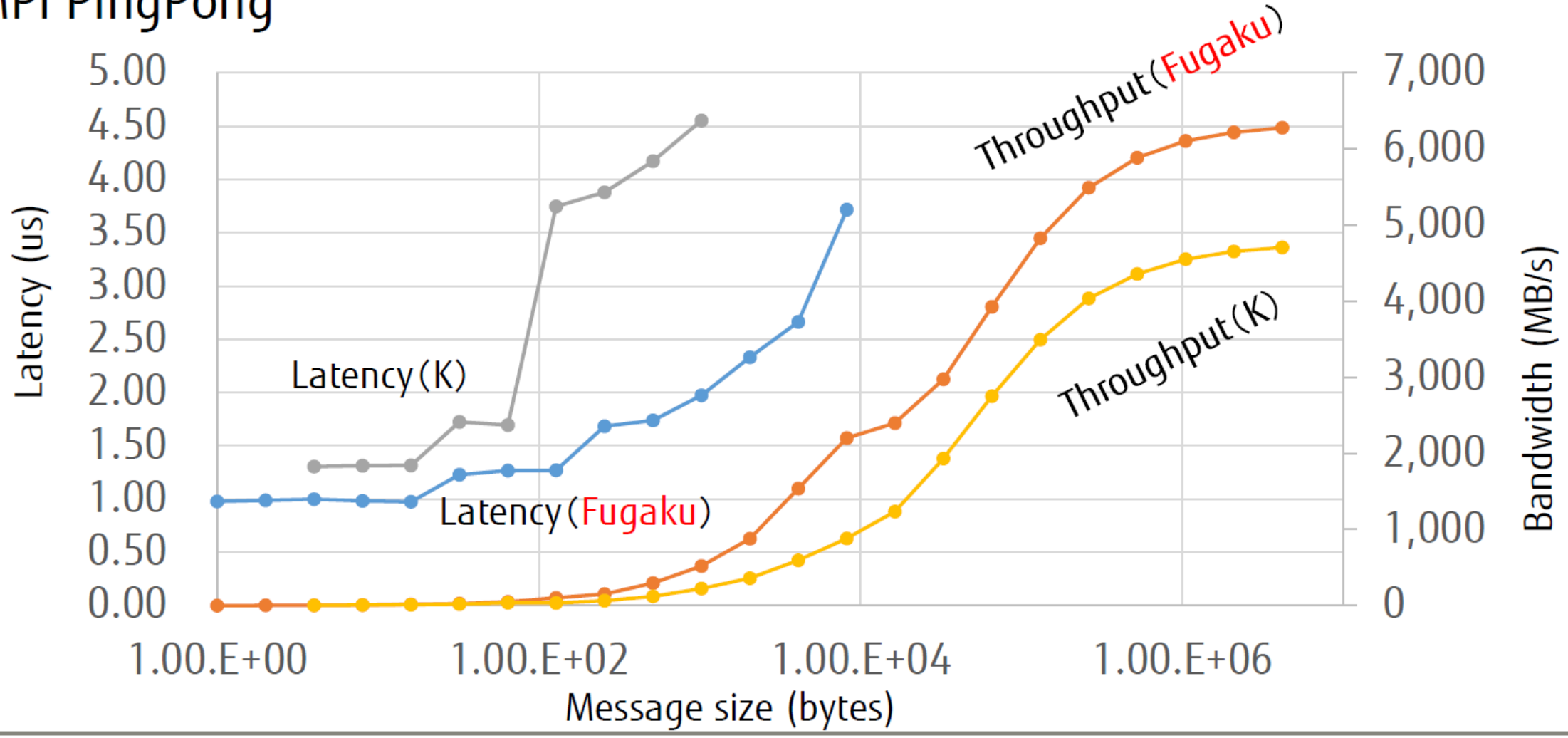
8B Put latency	0.49 – 0.54 usec
1MiB Put throughput	6.35 GB/s

rf. Yuichiro Ajima, et al. , “The Tofu Interconnect D,” IEEE Cluster 2018, 2018.

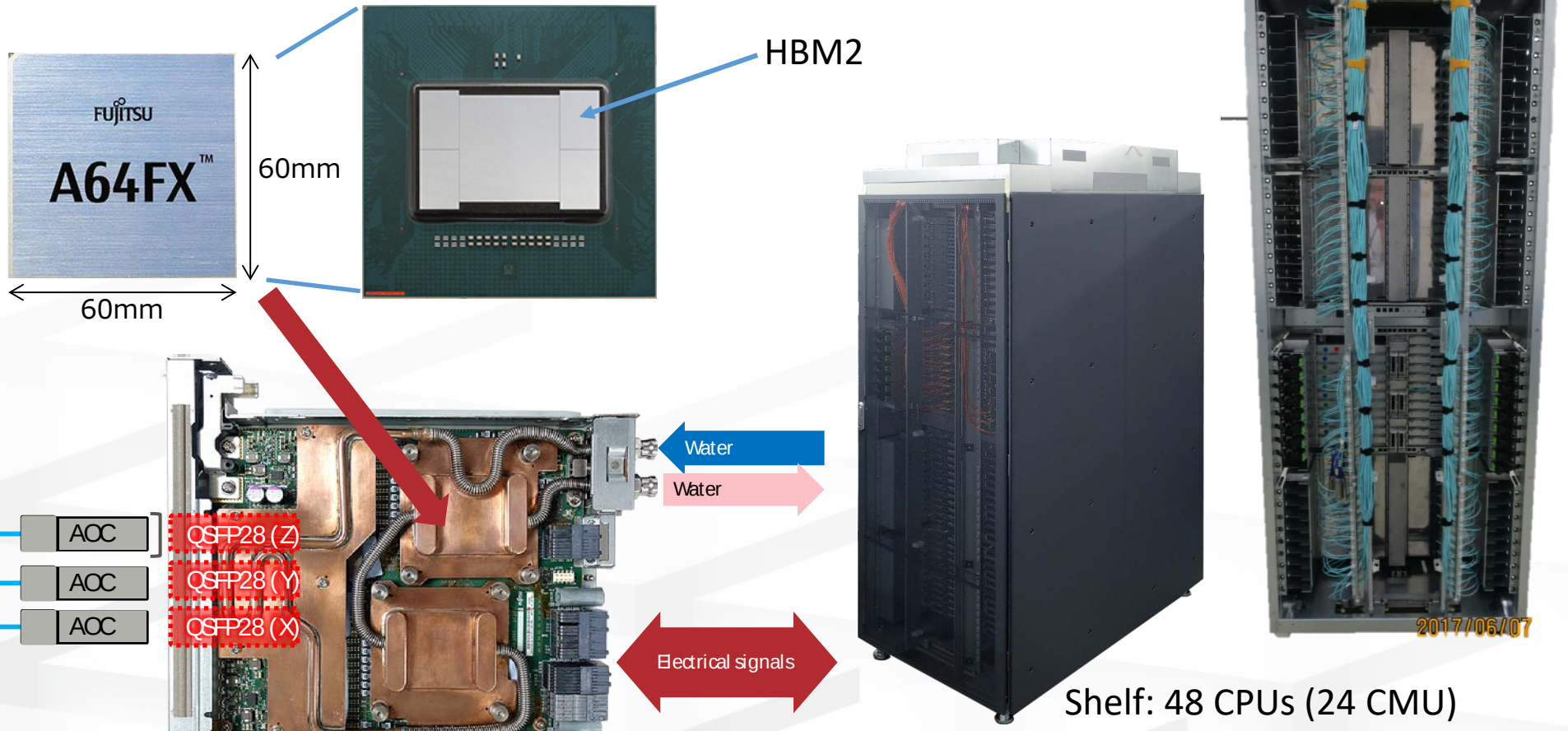
TofuD: MPI_Send/Receive Latency and BW

FUJITSU

MPI PingPong



Fugaku prototype board and rack



2 CPU / CMU

Shelf: 48 CPUs (24 CMU)

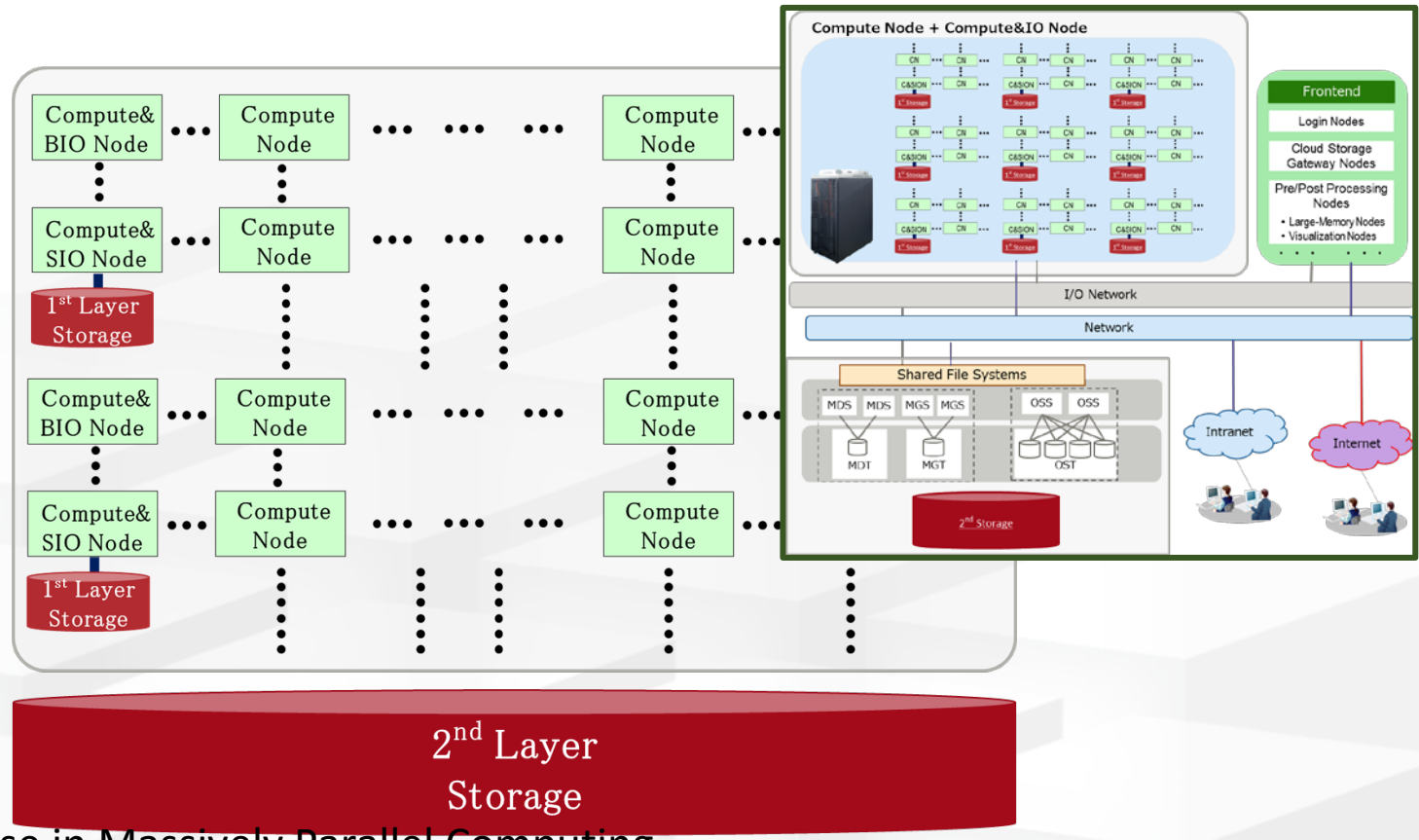
Rack: 8 shelves = 384 CPUs (8x48)

Fugaku System Configuration

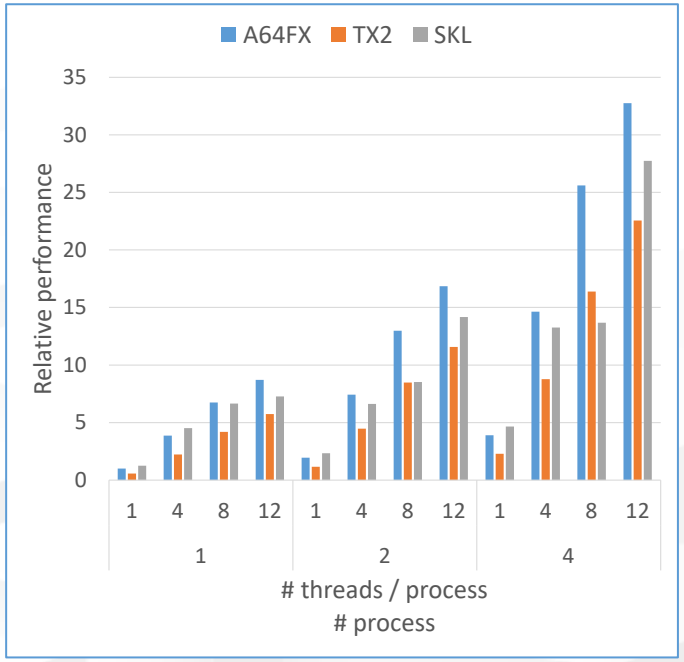
- **158,976 node**
- **Two types of nodes**
  - Compute Node and Compute & I/O Node connected by Fujitsu Tofu-D, 6D mesh/torus Interconnect
- **3-level hierarchical storage system**
  - 1st Layer
    - One of 16 compute nodes, called Compute & Storage I/O Node, has SSD about 1.6 TB
    - Services
      - Cache for global file system
      - Temporary file systems
        - Local file system for compute node
        - Shared file system for a job
  - 2nd Layer
    - Fujitsu FEFS: Lustre-based global file system, about 150 PB
  - 3rd Layer
    - Cloud storage services

Storage System

		Minimum Throughput	Measured Throughput
1st Storage	Write	49 MB/s /node	125 MB/s /node
	Read	113 MB/s /node	293 MB/s /node
2nd Storage	Write	200 GB/s /volume	220 GB/s /volume
	Read		211 GB/s /volume



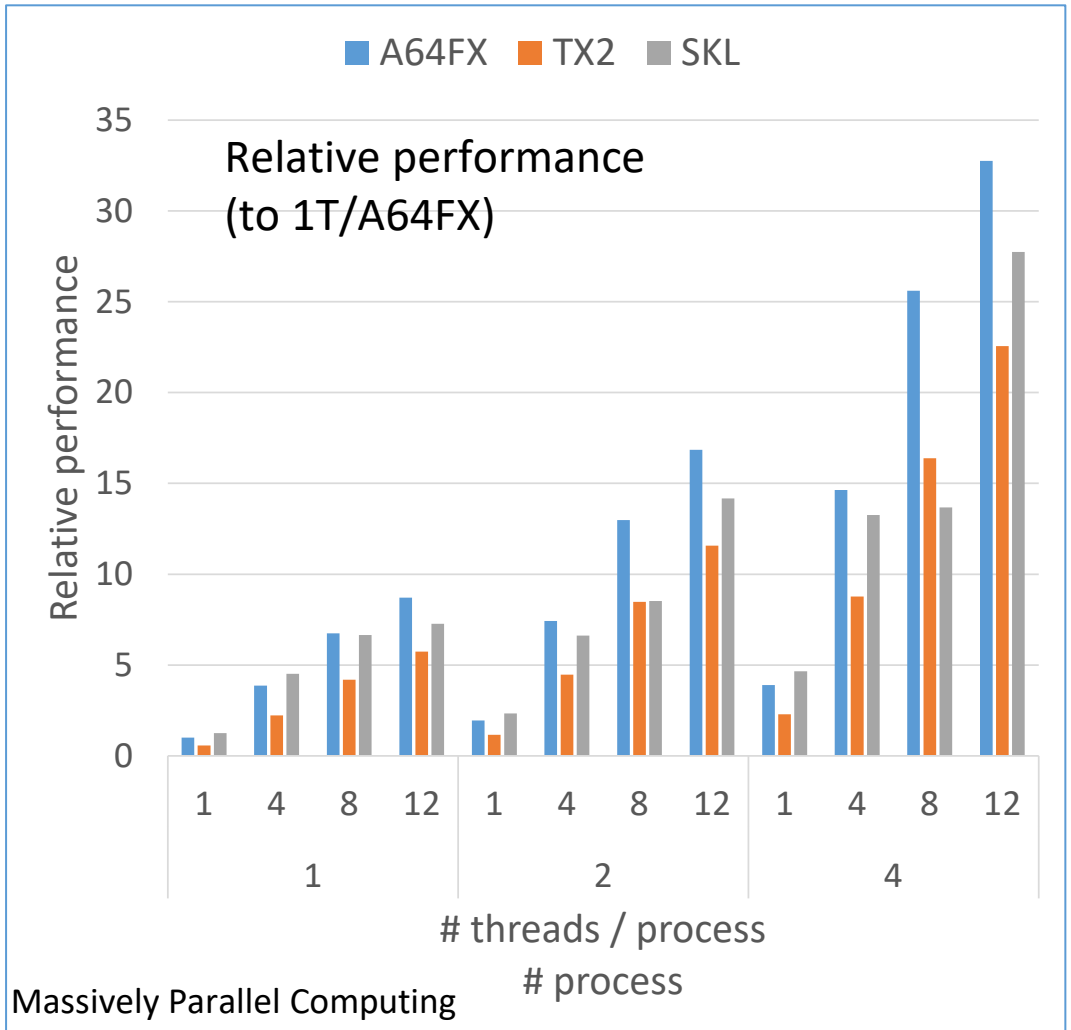
Performance of A64FX processor



Benchmark result of CloverLeaf

Taken form UK benchmarks:
A hydrodynamics mini-app to solve the compressible Euler equations in 2D, using an explicit, second-order method

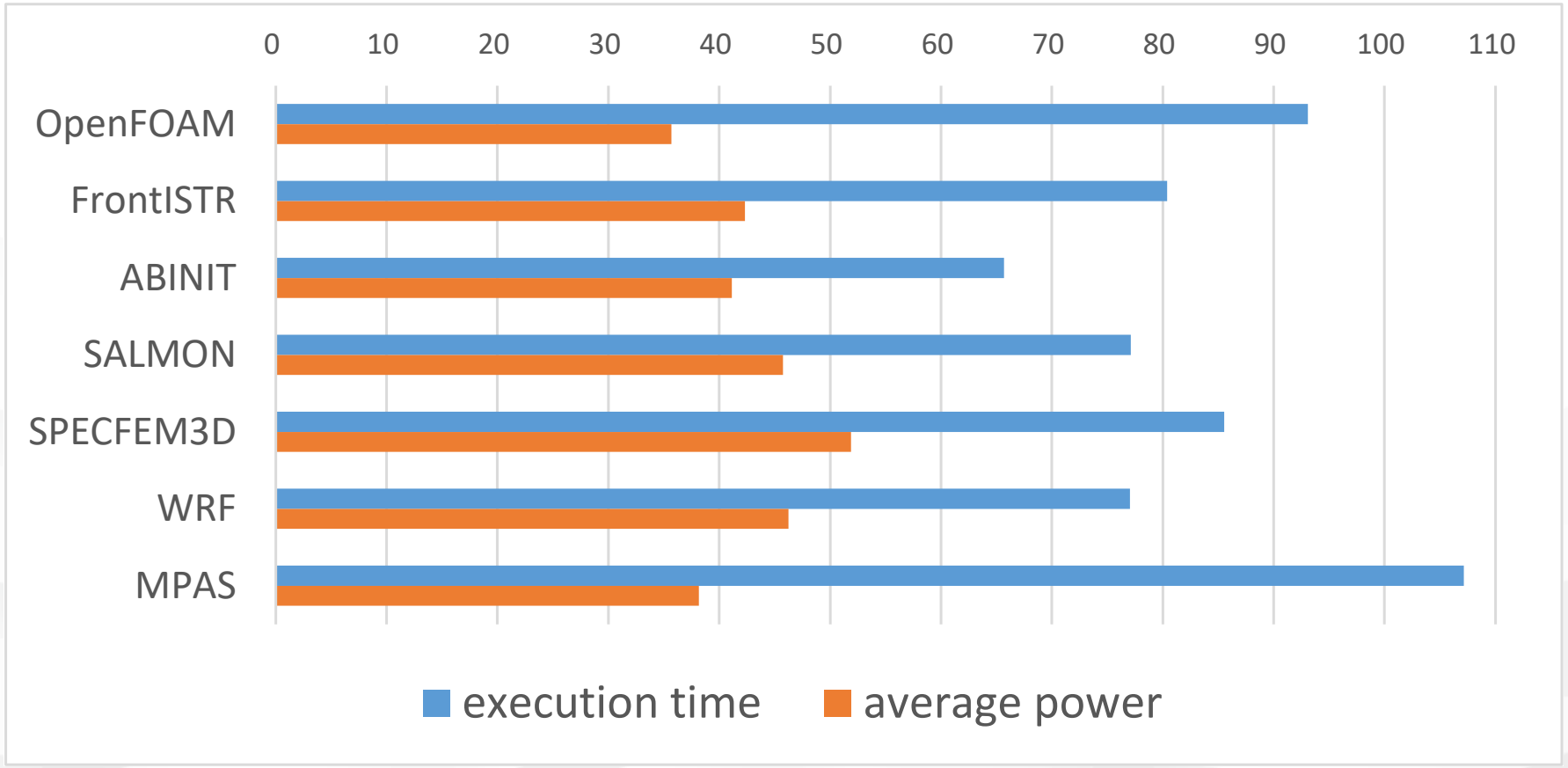
- Comparison with two nodes of TX2 (dual) and Skylake (dual)
- Good scalability by increasing the number of threads within CMG.
- The performance of one A64FX is comparable (better) to that of two nodes (4 sockets) of Skylake



Performance and Power-efficiency of HPC OSS

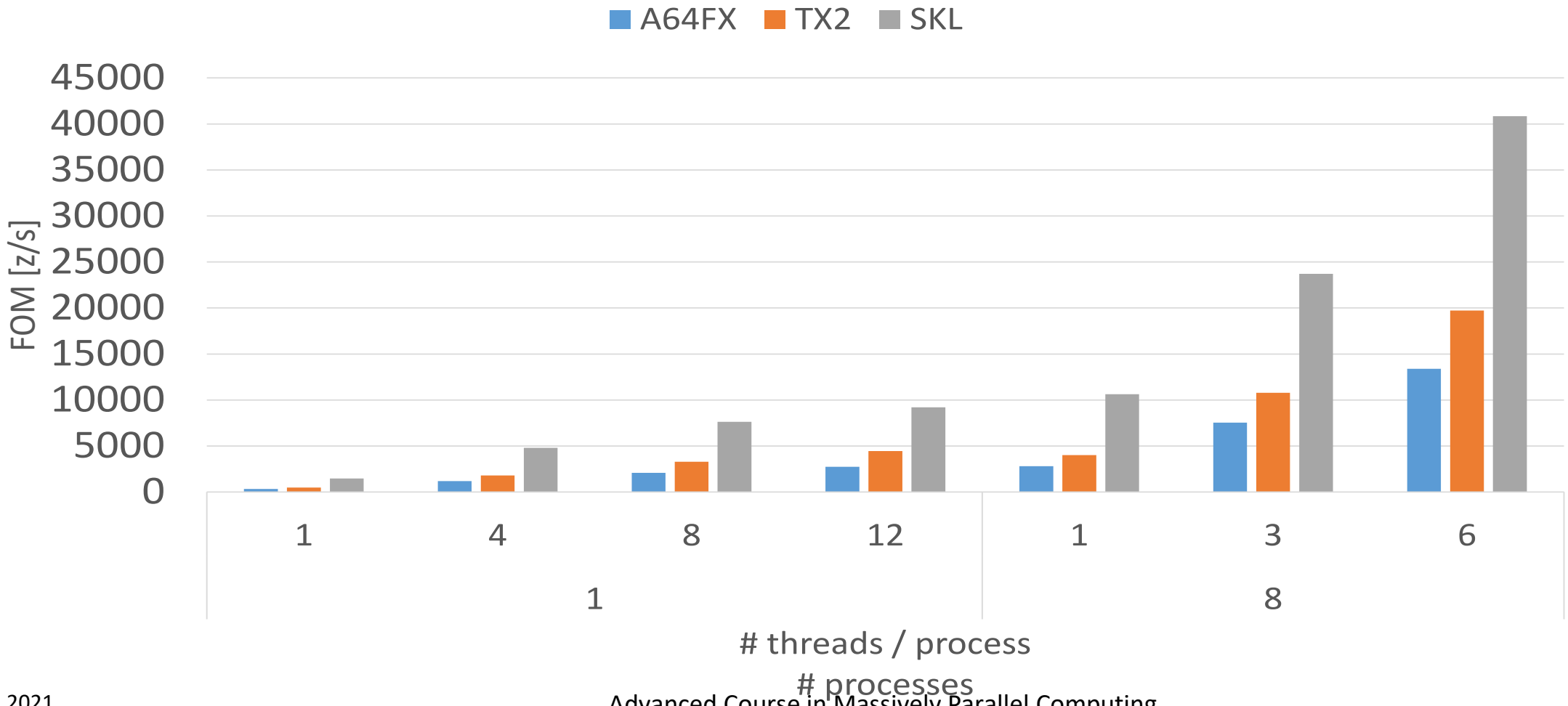
- **Several Open-source software were already ported and evaluated.**
- **Evaluation using one chip A64FX and dual chips of Xeon.**
- **The almost same performance to dual sockets of Xeon with half of power consumption.**

Performance and power efficiency of open-source applications
(results are shown in %, relative to Intel Xeon Platinum 8268 (Cascadelake, 2.90 GHz, 24 cores/socket) (dual sockets))



LULESH

- A64FX performance is less than Thx2 and Intel one
- We found low vectorization (SIMD (SVE) instructions ratio is a few %)
- We need more code tuning for more vectorization using SIMD



SPEC CPU® 2017 integer Speed

- **The performance of A64fX is about ¼ performance of Xeon in single thread.**
  - Fugaku uses normal mode (2.0GHz) with Fujitsu compiler tcsds-1.2.30a. For c and c++, clang mode is used.
  - Xeon is Cisco UCS B200 M5 (Platinum 8168(Skylake), 2.7GHz, 24core x 2 chip, turbo on) with icc 18.0.2.

https://www.spec.org/cpu2017/results/res2018q2/cpu2017-20180529-06367.txt

  - Reference machine is UltraSPARC-IV+(2.1GHz, 2cores x 4 chip)

- **The reason for the low single thread integer performance of A64FX is that**
  - the SIMD rate is low in SPEC CPU/int and
  - the frequency and the O3 resource are limited for the throughput-oriented architecture of A64FX.

	Lang	Threads	A64FX	Xeon
600.perlbench_s	C	1	1.20	6.20
602.gcc_s	C	1	2.63	9.57
605.mcf_s	C	1	3.42	11.2
620.omnetpp_s	C++	1	1.26	7.31
623.xalancbmk_s	C++	1	1.61	9.46
625.x264_s	C	1	2.06	11.6
631.deepsjeng_s	C++	1	1.37	5.17
641.leela_s	C++	1	1.26	4.36
648.exchange2_s	F90	1	1.42	13.2
657.xz_s	C/OpenMP	48	8.52	23.5
SPECspeed®2017_int_base			1.98	9.07

```
COPTIMIZE       = -Nclang -Ofast -mcpu=a64fx+sve -ffj-no-fp-relaxed -ffj-eval-concurrent -fsave-optimization-record -fopenmp -Nlst=t -Koptmsg=2
CXXOPTIMIZE     = -Nclang -Ofast -mcpu=a64fx+sve -ffj-no-fp-relaxed -ffj-eval-concurrent -fsave-optimization-record -fopenmp -Nlst=t -Koptmsg=2
FOPTIMIZE       = -Kfast,openmp -Nlst=t -Koptmsg=2
```

SPEC OMP® 2012

- **The performance of A64FX using 48 thread is about 65% performance of Xeon using 56 thread (28 cores).**
  - Fugaku uses normal mode (2.0GHz) with Fujitsu compiler tcsds-1.2.30a. For c and c++, clang mode is used.
  - Xeon is Cisco C240 M5 (Platinum 8280(Cascade Lake), 2.7GHz, 28core x 1chip, hyperthread on (56threads), turbo on) with icc 19.0.1.

https://www.spec.org/omp2012/results/res2019q2/omp2012-20190313-00172.txt

  - Reference machine is Sun Fire X4140 (AMD Opteron 2384, 2.7GHz 4core x 2chips)

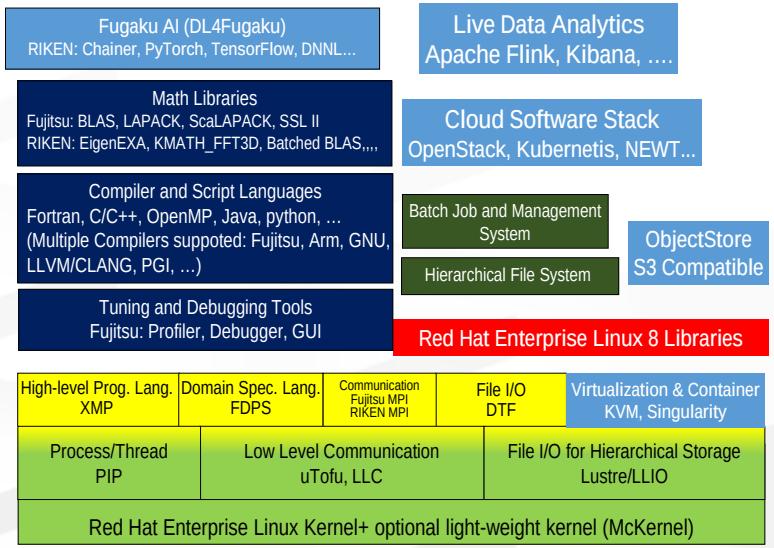
- **For some programs (swim and mgrid), A64FX brings extremely good performance due to HBM2.**
- **For 350.md, performance improvement has been confirmed by source code tuning, and we hope that it will be applied by improving the compiler.**

	Lang	Threads	A64FX	Xeon
350.md	F	48	2.63	62.6
351.bwaves	F	48	15.5	11.2
352.nab	C	48	3.00	12.9
357.bt331	F	48	5.82	16.0
358.botsalgn	C	48	5.22	10.5
359.botsspar	C	48	3.07	6.83
360.ilbdc	F	48	7.69	8.25
362.fma3d	F	48	4.28	11.3
363.swim	F	48	53.1	8.38
367.imagick	C	48	12.2	13.6
370.mgrid331	F	48	32.6	7.46
371.applu331	F	48	8.88	14.4
372.smithwa	C	48	12.8	11.8
376.Kdtree	C++	48	3.22	9.24
SPECompG_base2012			7.77	12.0

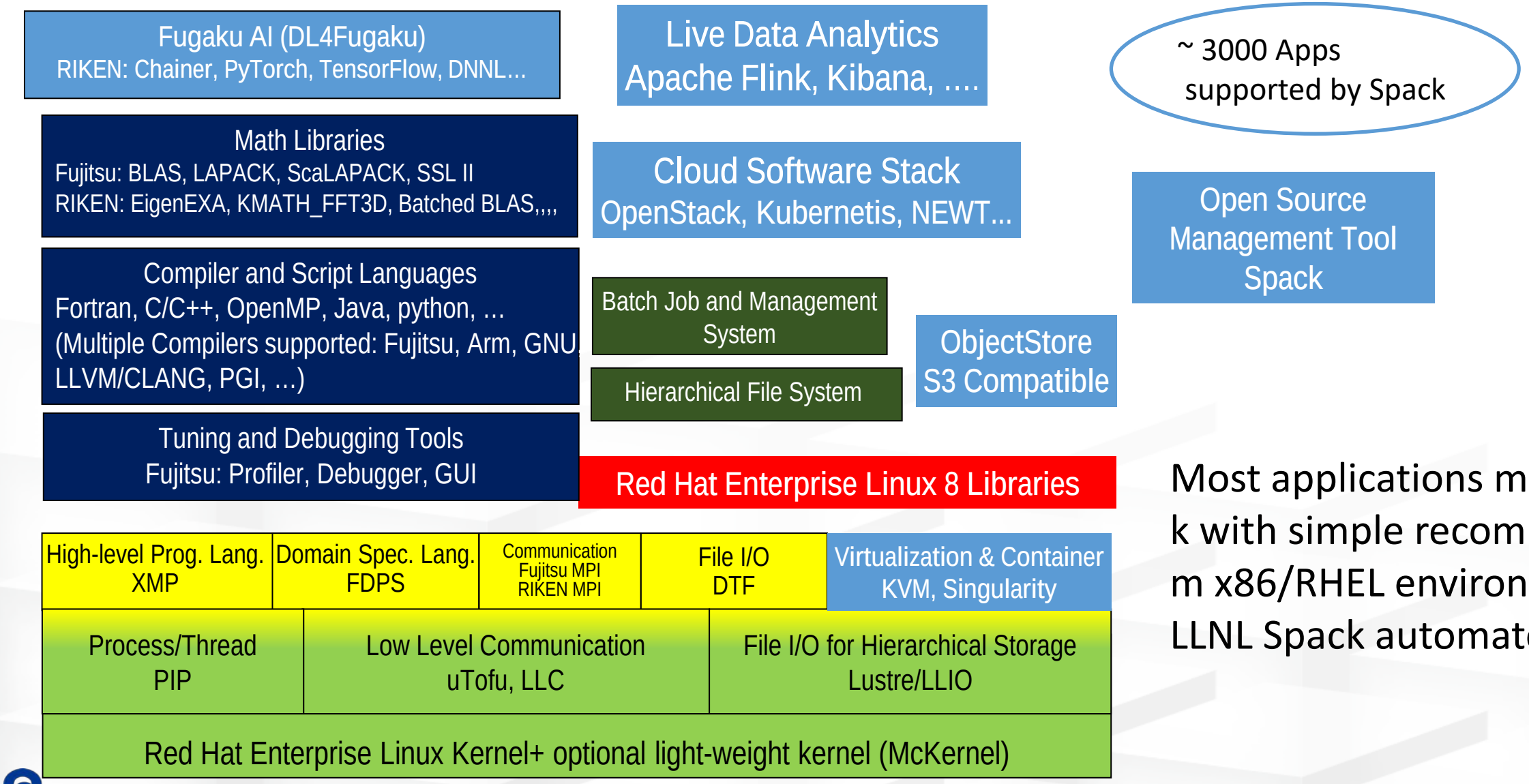
Summary of A64FX performance characteristics

- **For core-to-core comparison in intspeed, integer performance is ¼ of Xeon**
- **For chip-to-chip comparison in SPEC OMP, 48 threads performance of one chip is 65% to one chip of recent high-end Xeon (Cascade Lake)**
  - NOTE: Performance of memory-intensive benchmarks is extremely good in A64FX thanks to HBM.
- **For some scientific workload, the almost same performance to dual sockets of Xeon with half of power consumption (UK benchmark and HPC OSS)**
- **High SIMD rate is important to get performance**
  - Need to tune memory access pattern
  - We found many benchmark programs are not well-vecterized.
- **Power efficiency of A64FX is very good (double efficiency than Xeon?)**

System software overview of “Fugaku”



Fugaku System Software Stack



Most applications may wor
k with simple recompile fro
m x86/RHEL environment.
LLNL Spack automates this.

System software and Programming models & languages for “Fugaku”

- Standard programming model is OpenMP (for NUMA node(CMG)) + MPI
  - Both OpenMPI (by Fujitsu) and MPICH (by Riken) are supported.
  - 4 compilers (Fujitsu, gcc, LLVM/Arm, Cray), OpenMP 4.x is supported.
  - uTofu low-level comm. APIs for Tofu-D interconnect.
- Container and Virtual machine (KVM, Singularity, …)
- DL4Fugaku: AI framework for A64FX and Fugaku, used in Chainer, PyTorch, TensorFlow
- Many Open-source software are already ported using Spack

- System software and Programming tools, Math-Libs developed by RIKEN
  - McKernel: Light-weight Kernel enabling jitter-less environment for large-scale parallel program execution.
  - XcalableMP directive-based PGAS Language
  - FDPS: DLS for Framework for Developing Particle Simulators.
  - EigenExa: Eigen-value math library for large-scale parallel systems.

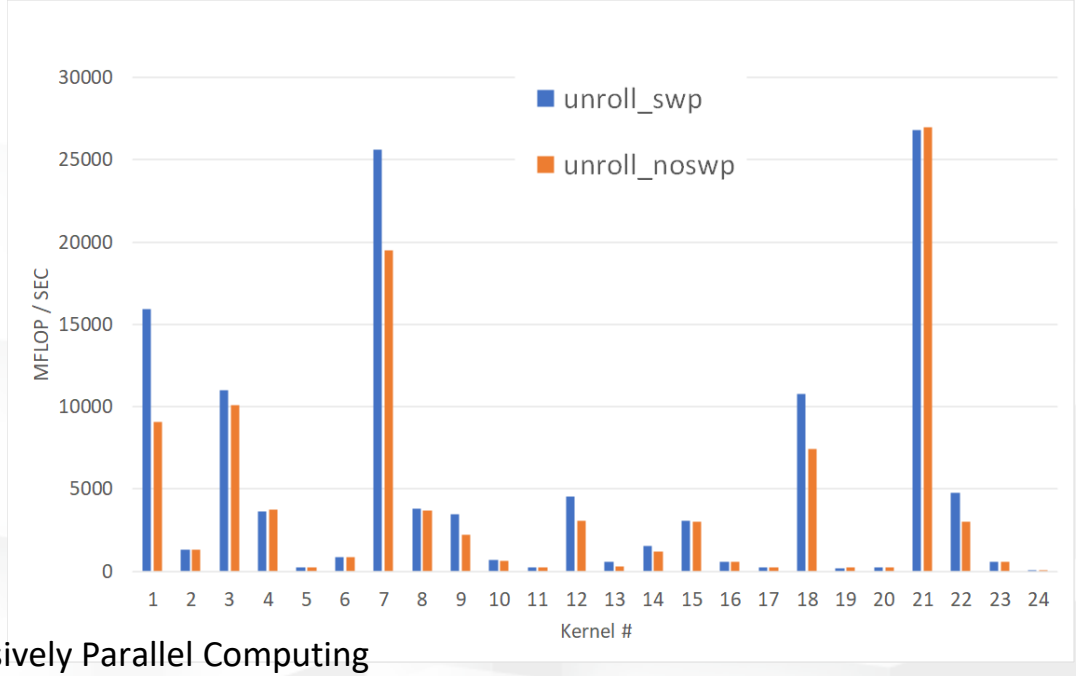
Performance Tuning for A64FX processor

- **HPC-oriented design**
  - Small core ⇒ Less O3 resources
  - (Relatively) Long pipeline
    - 9 cycles for floating point operations
    - Core has only L1 cache
  - High-throughput, but long-latency
  - Pipeline often stalls
    for loops having complex body.

	A64FX	Skylake
ReOrder Buffer	128 entries	224 entries
Reservation Station	60 (=10x2+20x2) entries	97 entries
Physical Vector Register	128 (=32 + 96) entries	168 entries
Load Buffer	40 entries	72 entries
Store Buffer	24 entries	56 entries

A64FX : https://github.com/fujitsu/A64FX
Skylake : https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)

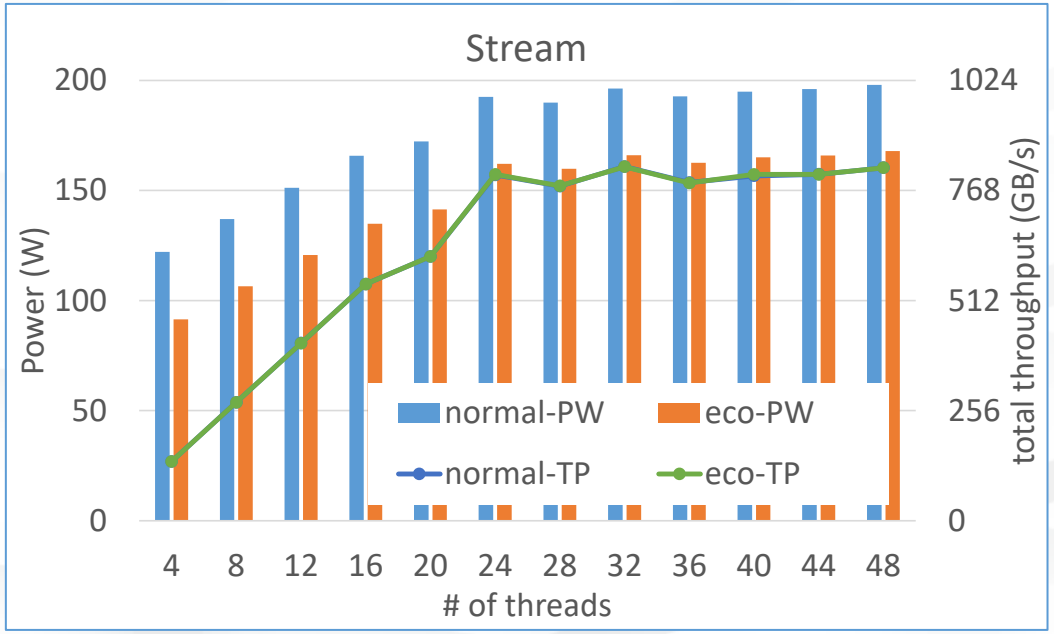
- **Compiler optimization (Fujitsu compiler)**
  - SWP: software pipelining
    - ～ 20% speedup in Livermore Kernels
  - Automatic and Manual loop fissions



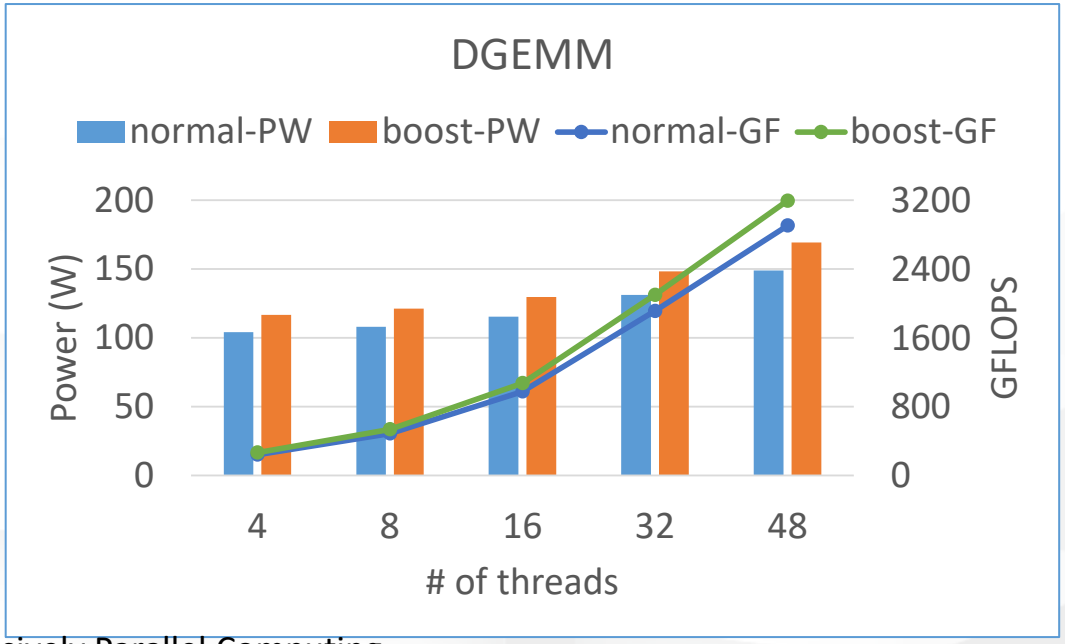
Performance improvement by SWP in Livermore Kernels by Fujitsu compiler

Evaluation power mode: Boost mode (2.2GHz) & Eco mode (1 SIMD pipeline)

- **Power & Performance of STREAM using Eco mode**
  - The performance is almost the same as that in normal mode (24 threads hits 80% of peak memory bandwidth
  - The power increases upto 24 threads.
  - 15%-25% reduction comparing to that in normal mode.



- **Power & Performance of DGEMM (in Fujitsu Lib) using Boost mode**
  - Reach to 95% out of peak performance
  - The performance is 10% better than that in normal mode.
  - The power increases by 13.7%
  - The power-efficiency decreases by 3.3 %



スーパーコンピュータ「富岳」

432 racks
158,976 nodes

Half exaflops in DP
1 Exaflops in SP!

2019年12月から搬入開始
2020年5月に搬入終了
６月にベンチマークで世界一
2021年3月から共用開始

https://www.r-ccs.riken.jp/fugaku/3d-models/

「京」から「富岳」への進歩

	京	富岳	比
ノードのコア数	8	48	
半導体技術　(nm)	45	7	
コア当たりの性能(GFLOPS)	16	64 (70)	4(4.4)
ノード当たりの性能. (TFLOPS)	0.128	3.07 (3.37)	24 (26.4)
ノードのメモリバンド幅(GB/s)	64	1024	
ラック当たりのノード数	96	384	4
ラックの性能(TFLOPS)	12.3	1180 (1297)	95(105)
システム全体のノード数	82,944	158,976	
システム性能（PFLOPS)倍精度	10.6	488 (537)	42.3 (52.2)
単精度	10.6	977 (1070)	84.6 (104.4)

「ノード」とは、ネットワークにつながるコンピュータの単位
富岳の場合は、ノードは１チップからなる。１チップは、複数のコア（コンピュータ）を内蔵する。

TOP500, HPCG, HPL-AI, Graph500

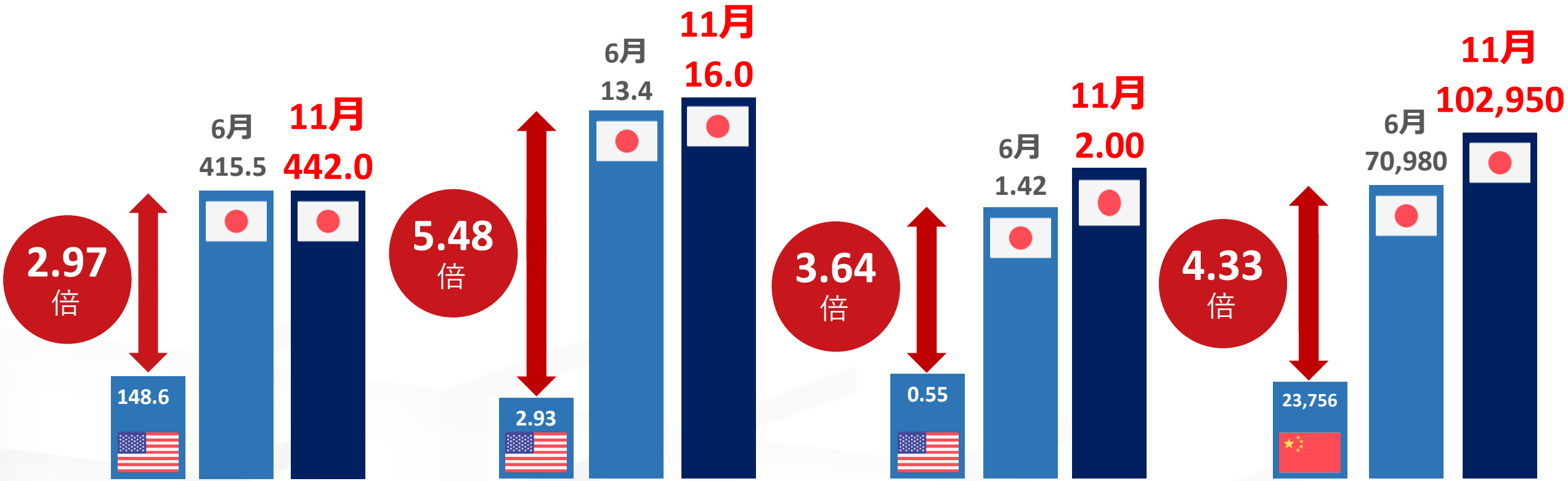
2020年6月時点

	ノード数	周波数 (GHz)	測定値	ピーク性能	効率	使用ノード数の割合	第2位との比較	
							性能	性能差（倍率）
TOP500	152,064	2.2	415.53 PF	513.85 PF	80.9%	96%	148.60 PF	2.8
HPCG	138,240	2.2	13.36 PF	467.14 PF	2.9%	87%	2.92 PF	4.6
HPL-AI	126,720	2.0	1.42 EP	1.55 EP	91.3%	80%	0.55 EF	2.5
Graph500	92,160	2.2	70.98 Tteps			58%	23.75 Tteps	3.0

備考：性能値は小数点第3位以下切り捨て

- **TOP500**
  - 密行列係数連立1次方程式をLU分解法で求解するプログラム(Linpack)を使用
- **HPCG**
  - 疎行列係数連立一次方程式を共役勾配法（conjugate gradient）で求解するプログラムを使用
- **HPL-AI**
  - 密行列係数連立1次方程式をLU分解かつ混合演算（倍精度浮動小数点(FP64)、半精度浮動小数点(FP16))によって求解するプログラムを使用
- **Graph500**
  - グラフ探索問題

「富岳」全系によるベンチマークテスト結果（6月の結果との比較）



TOP500
（単位：PFLOPS）
浮動小数点の演算での性能評価

HPCG
（単位：PFLOPS）
実際にアプリケーションを稼働させた性能評価

HPL-AI
（単位：EFLOPS）
AI処理での性能評価

Graph500
（単位：GTEPS）
ビッグデータ処理での性能評価

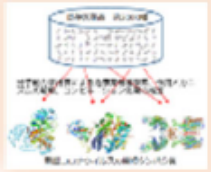
2位に対して、3倍から5.5倍近い性能差を実現

2020年6月22日発表
International Supercomputing Conference (ISC 2020)　スーパーコンピュータ世界ランキング

昨年度、共用前に前倒しで実施されたコロナ対策のための計算科学

医学的側面からの研究

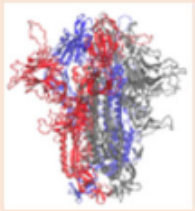
「富岳」による新型コロナウイルスの治療薬候補同定



分子動力学計算により、約2000種の既存医薬品の中から、新型コロナウイルスの標的タンパク質に高い親和性を示す治療薬候補を探索・同定する。

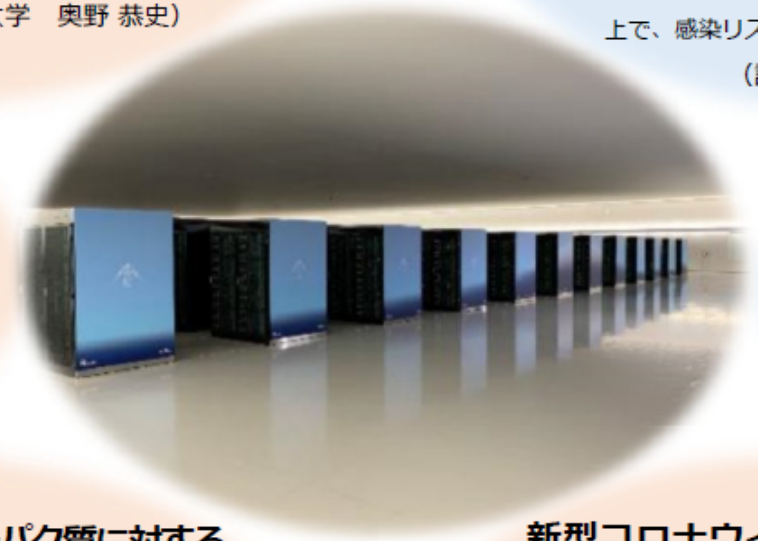
（課題代表者；理化学研究所/京都大学　奥野 恭史）

「富岳」を用いた新型コロナウイルス表面のタンパク質動的構造予測

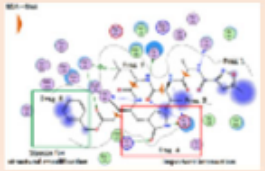


クライオ電子顕微鏡によって解かれたウィルス表面タンパク質の立体構造を初期モデルとして、その立体構造の動きを「富岳」を用いた分子動力学計算で予測する。

（課題代表者；理化学研究所　杉田 有治）



新型コロナウイルス関連タンパク質に対するフラグメント分子軌道計算



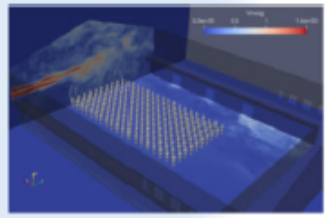
新型コロナウイルス関連タンパク室に対するフラグメント分子軌道計算を系統的に実施し、詳細な相互作用解析を行う。

（課題代表者；立教大学　望月 祐志）

社会的側面からの研究

室内環境におけるウイルス飛沫感染の予測とその対策

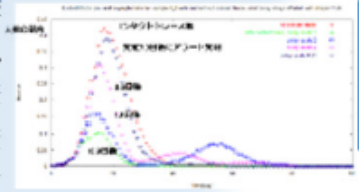
通勤列車内、オフィス、教室、病室といった室内環境において、新型コロナウイルスの特性を考慮した飛沫の飛散シミュレーションを行い、感染リスク評価を行った上で、感染リスク低減対策の提案を行う。



（課題代表者；理化学研究所/神戸大学　坪倉 誠）

パンデミック現象および対策のシミュレーション解析

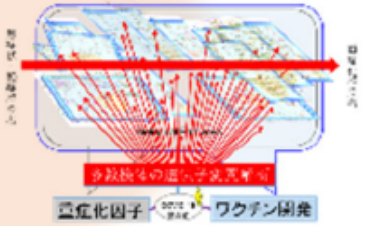
今後生じうる社会経済活動への影響を評価し、収束シナリオとその実現方法を探る。あわせてウイルスの変異などにより感染・発病の経過が変化した場合に起こりうる事象への対応を立案する。



（課題代表者；理化学研究所　伊藤 伸泰）

新型コロナウイルス感染症重症化に関するヒト遺伝子解析

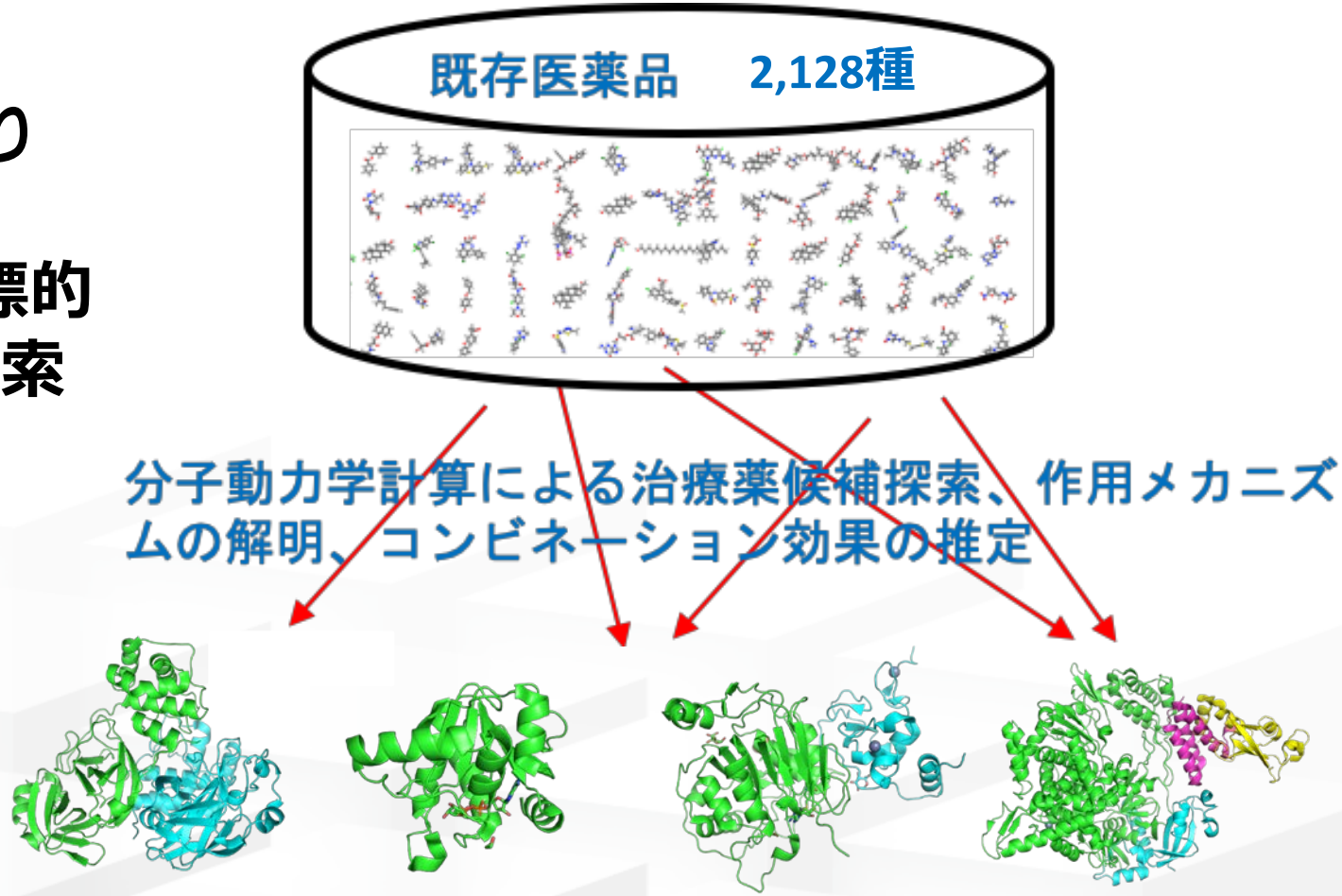
新型コロナウイルスの重症化例および軽症ないし無症状感染例について、全ゲノムシーケンスを用いた解析を実施し、スパコンシミュレーションによる重症化リスク関連遺伝子変異を同定する。



（課題代表者；東京医科歯科大学　宮野 悟）

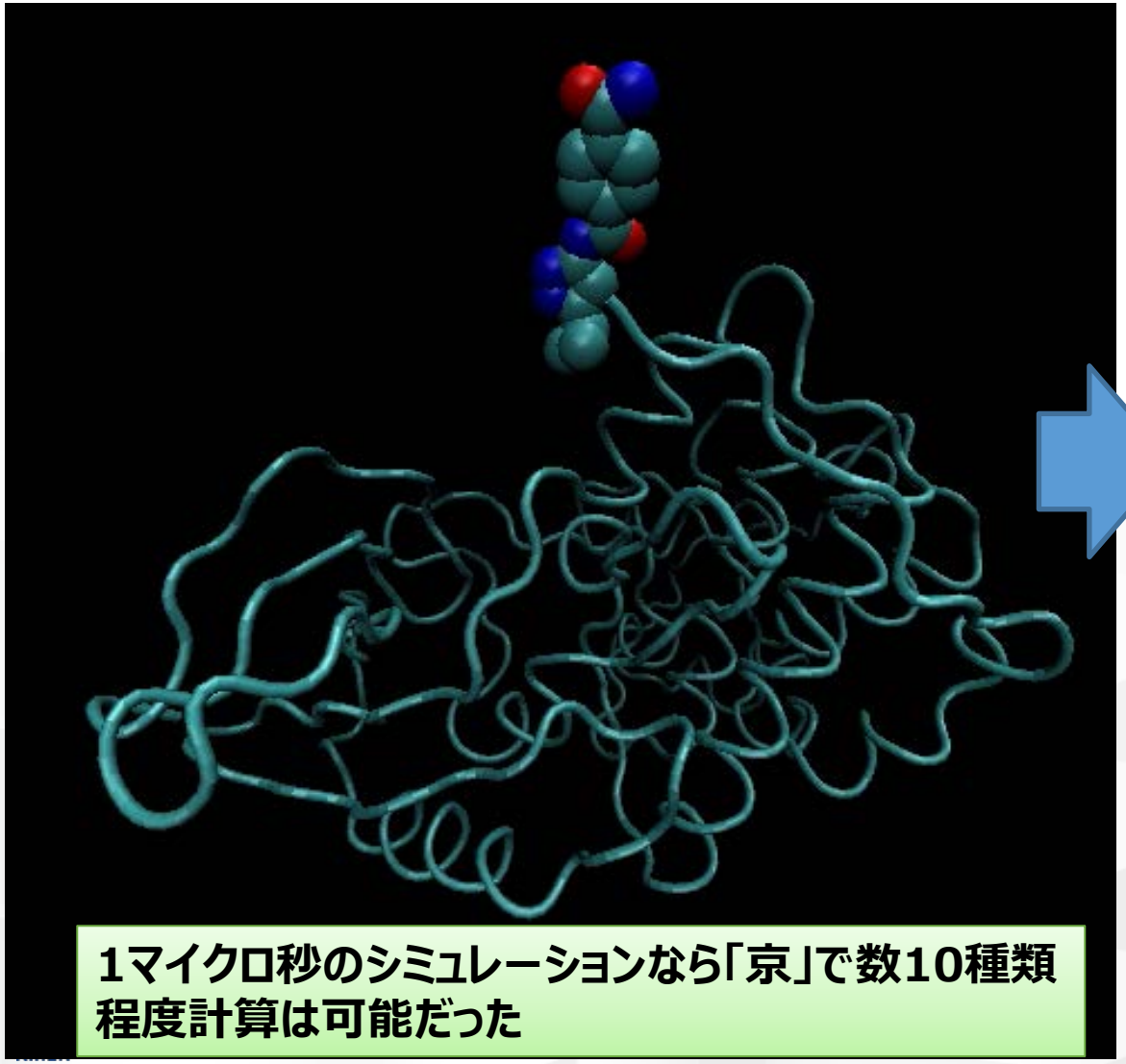
「富岳」による新型コロナウイルスの治療薬候補同定

「富岳」を用いた分子シミュレーション（MD計算：分子動力学計算）により2,128種の既存医薬品の中から新型コロナウイルスの増殖に関連する標的タンパク質に作用する治療薬候補を探索する。



スーパーコンピュータ「富岳」でのチャレンジ

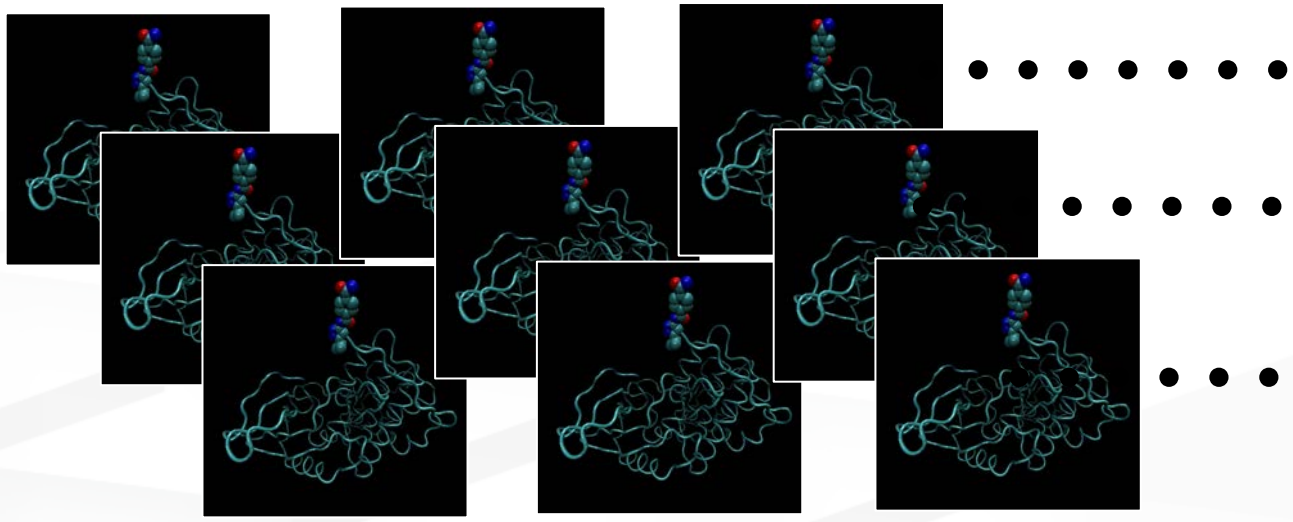
薬剤がタンパク質にどのように結合するかを
タンパク質を動かしながら計算（MD計算）



1マイクロ秒のシミュレーションなら「京」で数10種類程度計算は可能だった

医薬品開発では、多くの化合物候補から、タンパク質に結合する化合物を探索する必要がある

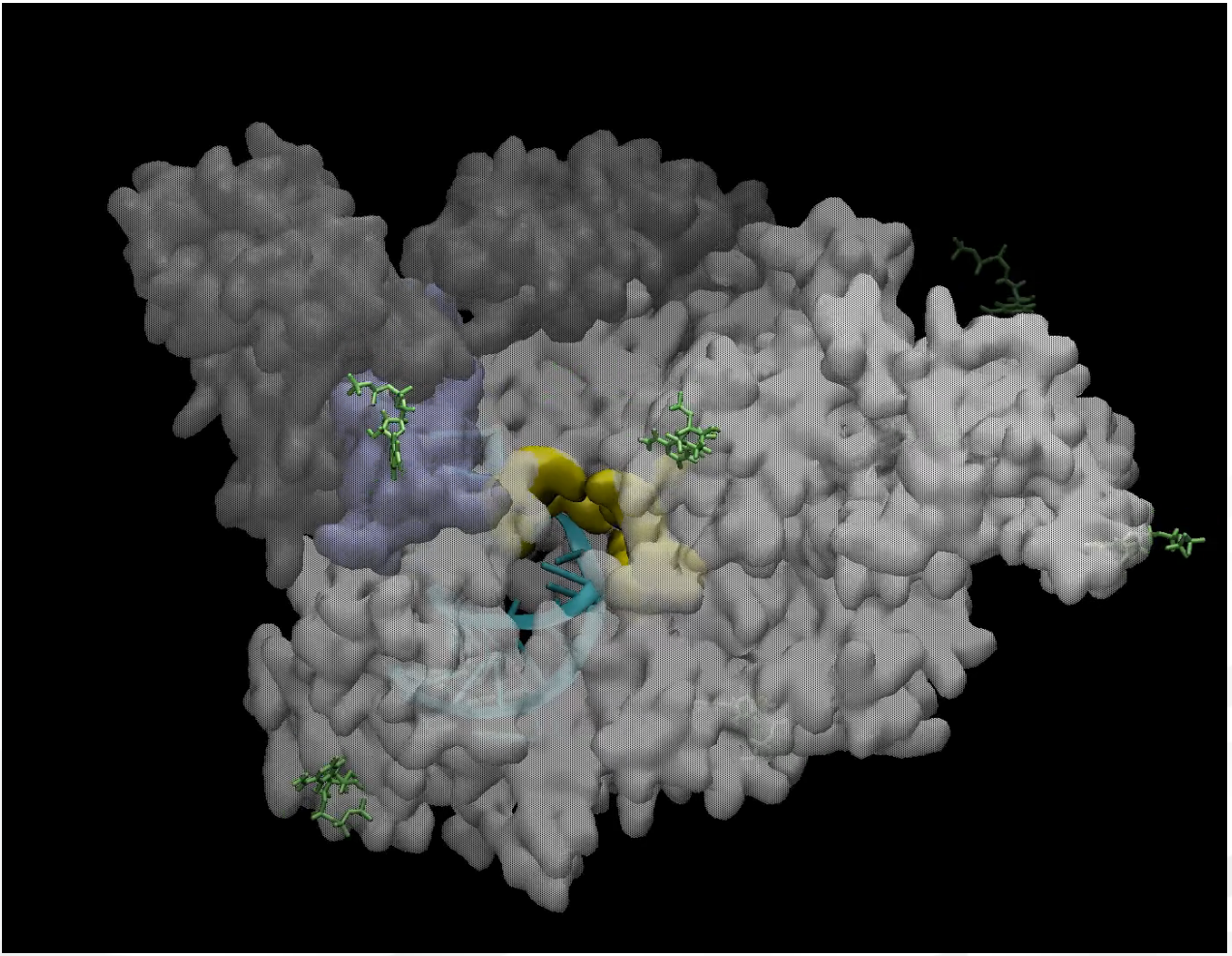
「富岳」で数1000種類の化合物に対する、タンパク質との結合のシミュレーションを実現する



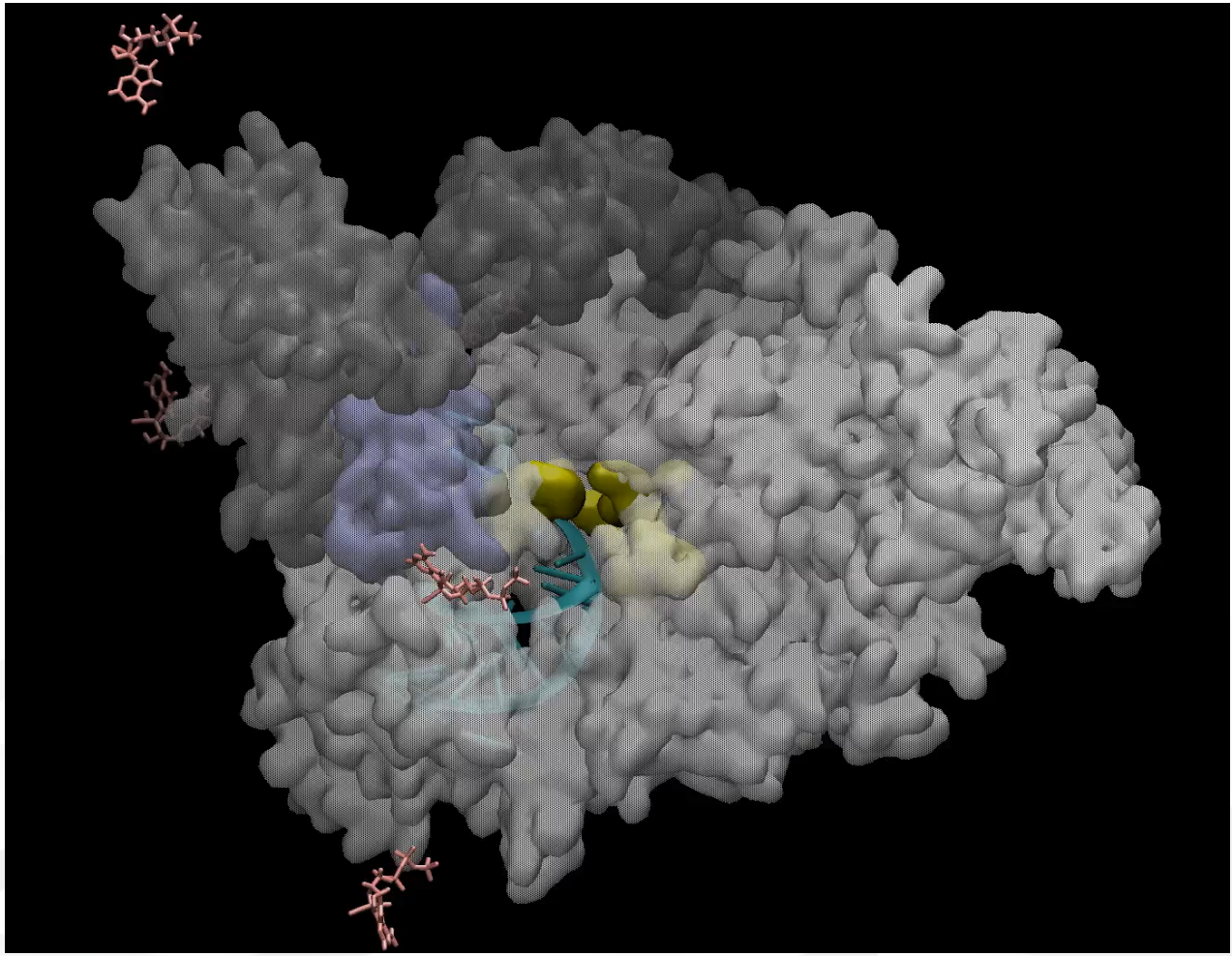
成功すれば、世界初の成果
→ 高精度で多くの化合物の結合が評価できるようになると実際に実験する手間が劇的に削減できる

アビガン、レミデシブルとRNAポリメラーゼ（RdRp）との作用の様子

アビガン（緑色）の結合シミュレーション



レミデシブル（桃色）の結合シミュレーション



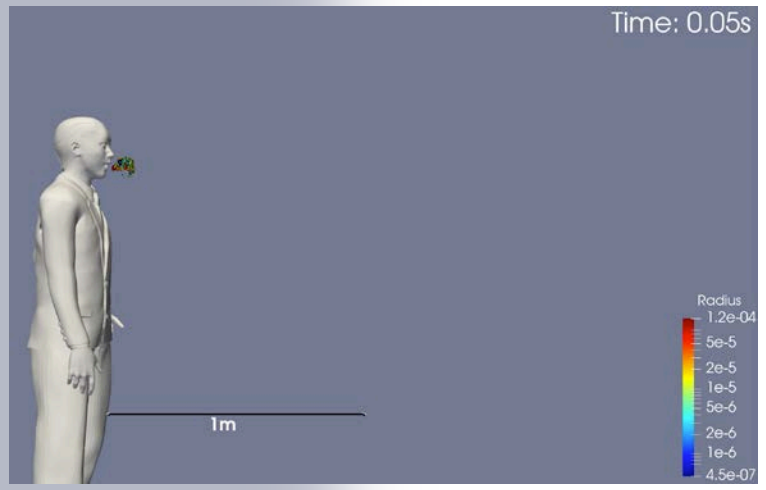
灰色：標的タンパク質であるRNAポリメラーゼ（RdRp）　黄色：RNAポリメラーゼの活性ポケット

「富岳」による飛沫・エアロゾル感染シミュレーション

感染症に対する非科学的な恐れ，
根拠のない過信と侮り

新型コロナに対する圧倒的な
科学的データ不足

圧倒的な計算資源を活用した高精度シミュレーションによる科学的データの提示と飛沫の見える化による現象の理解



社会に対する飛沫感染の正しい理解とその予防の啓発

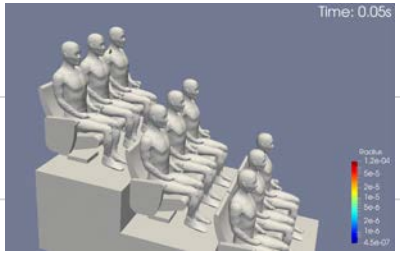
テレビ・ラジオ：300件，新聞：275件
ウェブ記事：1150件

**行政機関や各種業界との連携による
ガイドラインの策定や改定**

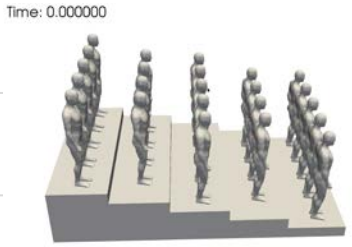
スマートライフ実現のためのAI等を活用したシミュレーション調査研究（内閣官房）

「富岳」だからこそ対応できた高精度・高速・多ケース解析

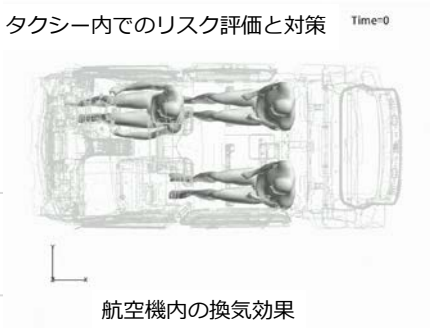
コンサートホールでのリスク



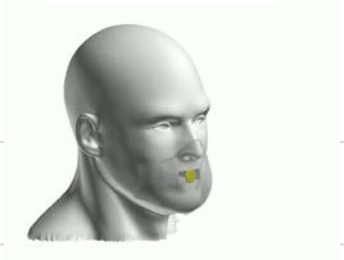
合唱活動での対策提案



タクシー内でのリスク評価と対策

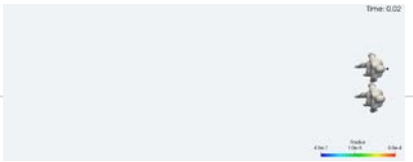


ウレタンマスクの問題

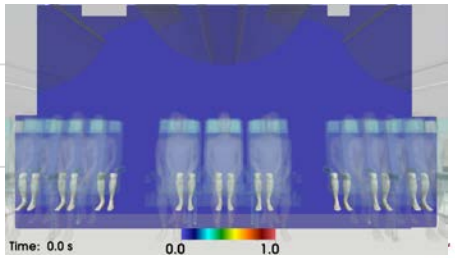


二重マスクの効果

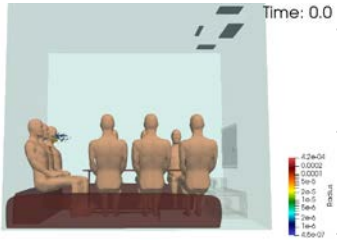
歩行時のソーシャルディスタンス



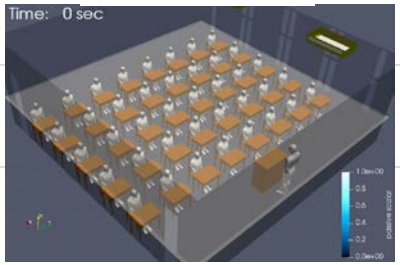
航空機内の換気効果



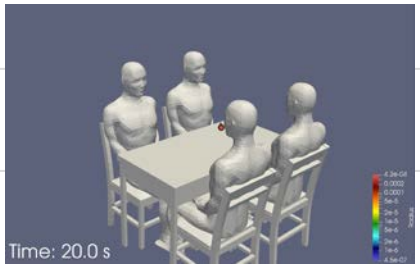
カラオケ店での対策提案



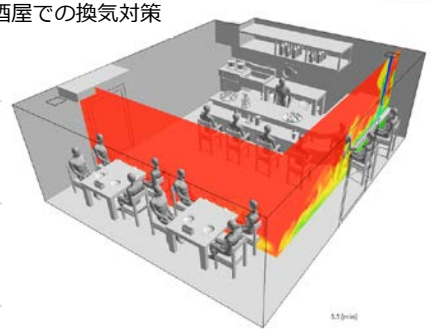
小学校での窓開け換気



飲食時のリスク評価

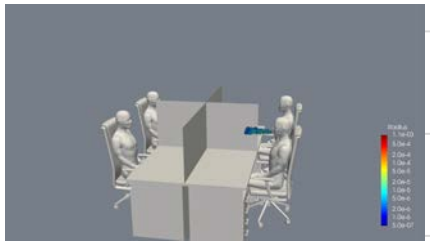


居酒屋での換気対策



路線バスでのエアコン換気の効果

オフィスでのパーティションの効果



満員電車内での窓開け換気の効果

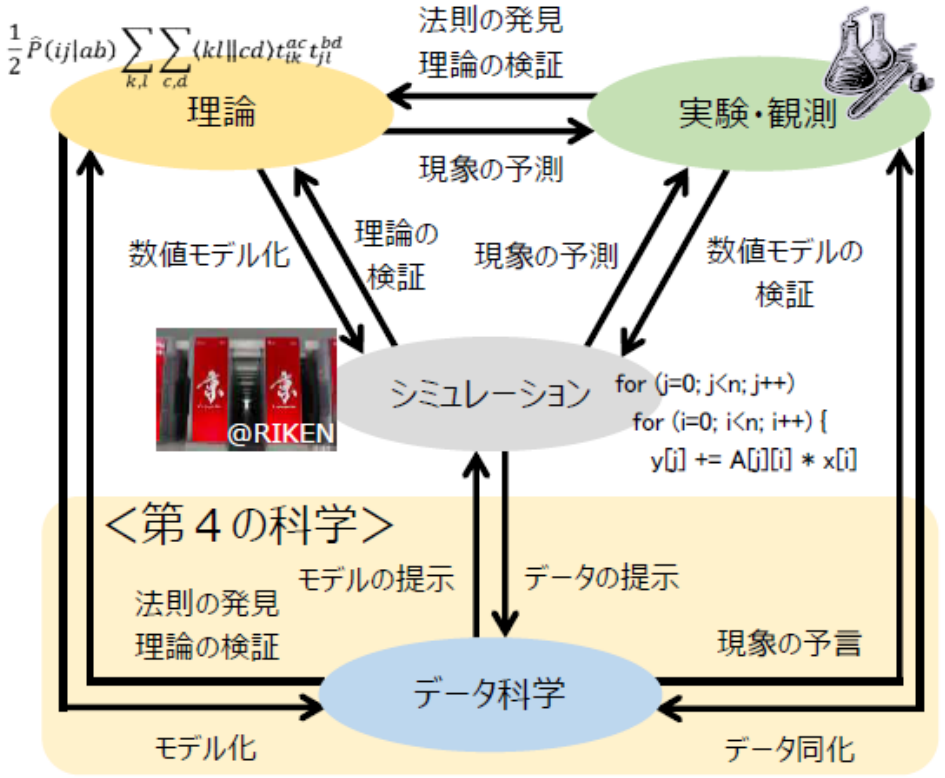
富岳への期待と
これからのスパコンの課題

富岳でのアプリケーション

- 京では、システム全体でなにができるか（capability)を目標としていた
- 富岳では、京で数ジョブしか同時に動かすことができなかった大規模アプリを数十から数百ジョブを同時に動かすことができるようになる。（capacity)
  - 実際の科学の研究では、いろいろなケースを試してみることが大切

- シミュレーションと
  データ科学の融合

- AIを加速する仕組みも！

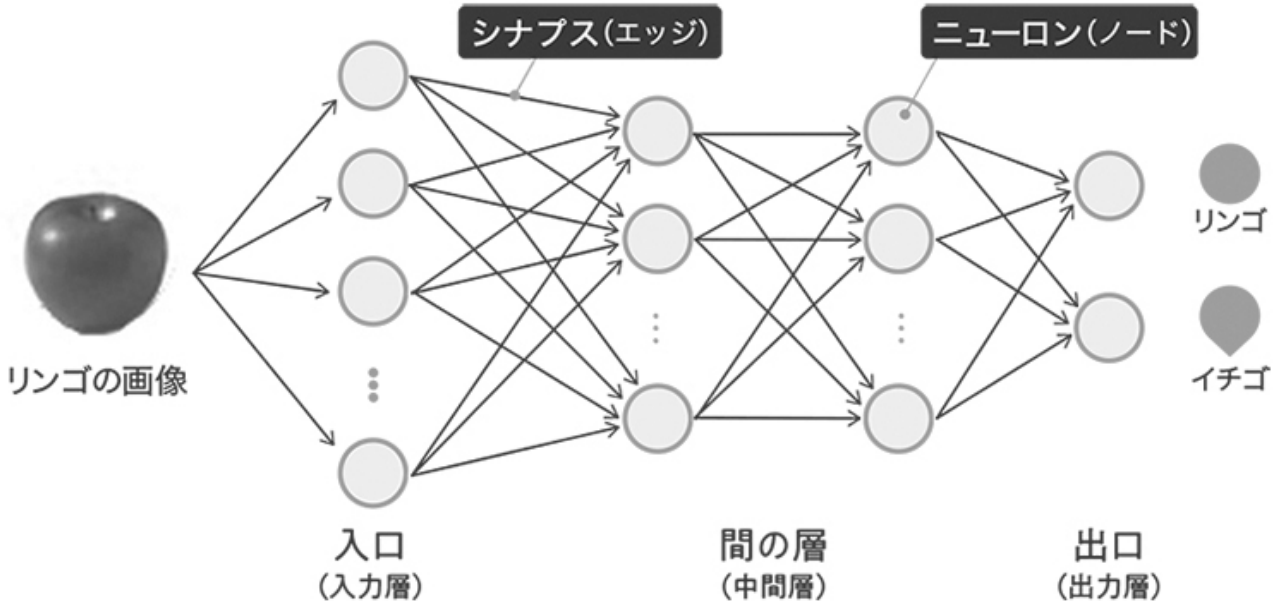


文科省のホームページより　http://www.mext.go.jp/b_menu/shingi/chousa/shinkou/020/shiryo/__icsFiles/afieldfile/2018/09/18/1409147_4_1.pdf

スパコンとAIの関係

AI（ディープラーニング）とは？

- ディープラーニングは、現在、AIの主流となっている方法。多層のニューラルネットワークを使ったディープニューラルネットワーク（DNN）を用いた学習
- 学習(training)とは、「シナップス（エッジ）」の強さを決める作業。それを使う動作を推論（inference)という。
- 十分なデータ量があれば、人間の力なしに機械が自動的にデータから特徴を抽出してくれるので注目されている。



https://internet.watch.impress.co.jp/img/iw/docs/1138/111/html/ai01-01.jpg.html

スパコンとAI(ディープ・ラーニング)

- ディープラーニングでは、大量のパータンを読み込ませて、「学習」する必要があるために、大量の計算が必要。
- アプリごとに別の学習が必要

スパコンやGPUが必要。

- 富岳は、ディープラーニングの学習のための演算を高速化する命令がある。
- また、ディープラーニングのニューラルネットワークが大きくなると並列処理が必要

シミュレーションとAI（ディープ・ラーニング）の関係

- **科学技術計算とAI（ディープ・ラーニング）**
  - 時間のかかる計算の代わりに、あらかじめ学習をしておいたディープ・ラーニングで答えを求める

- **ディープ・ラーニングは、教え込んだ答えを出すだけであって、新しい答えが出せるわけではないので注意！**

- **学習のデータが足りない場合に、シミュレーションした結果を学習に使うことが期待されている。**

これからのスパコンの課題

- **電力の限界**
  - これ以上の性能を出すためには電力を増やす必要があるが、電力が限界
  - 米国（中国）のスパコンは特別な演算をするための仕組み（GPUなど）を使い、電力を効率化しているが、すべてのアプリに対応できなかったり、プログラムを書き換えないといけない。

- **スパコンの進歩に必要な半導体技術の限界が指摘されている**
  - 「ムーア」の法則の終焉、「ポストムーア」、新規デバイス
  - 計算原理の限界？　⇒　量子コンピューティング？

- **ビックデータ技術やAI技術との融合**
  - 比較的大きいシミュレーションをたくさんのケースを実行できるようになるので、データの処理やデータ処理との融合が必要
  - AIという新たな市場、あるいはAIという新たな計算技術

これからのスパコン（とプログラミングモデル）

The HPC Hardware Landscape

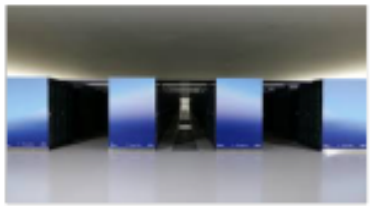
Current Generation: Programming Models OpenMP 3, CUDA and OpenACC depending on machine

LANL/SNL Trinity
Intel Haswell / Intel KNL
OpenMP 3

LLNL SIERRA
IBM Power9 / NVIDIA Volta
CUDA / *OpenMP*[a]

ORNL Summit
IBM Power9 / NVIDIA Volta
CUDA / *OpenACC* / *OpenMP*[a]

SNL Astra
ARM CPUs
OpenMP 3



Riken Fugaku
ARM CPUs with SVE
OpenMP 3 / *OpenACC*[b]

Upcoming Generation: Programming Models OpenMP 5, CUDA, HIP and DPC++ depending on machine

NERSC Perlmutter
AMD CPU / NVIDIA GPU
CUDA / *OpenMP 5*[c]

ORNL Frontier
AMD CPU / AMD GPU
HIP / *OpenMP 5*[d]

ANL Aurora
Xeon CPUs / Intel GPUs
DPC++ / *OpenMP 5*[e]

LLNL El Capitan
AMD CPU / AMD GPU
HIP / *OpenMP 5*[d]