Do-it-yourself Computational Astronomy — Hardwares, Algorithms, Softwares, and Sciences

Jun Makino
Center for Computational Astrophysics/
Division Theoretical Astronomy
National Astronomical Observatory of Japan

November 23, 2007

Caveats:

- **This lecture is NOT on mathematics.**

Caveats:

- This lecture is NOT on mathematics.
- I do not quite understand why the selection committee asked me to give lecture.

Caveats:

- This lecture is NOT on mathematics.
- I do not quite understand why the selection committee asked me to give lecture.
- I do not really know what should I talk.

Who am I?

- My name is Jun Makino (Junichiro Makino ...).
- Professor at National Astronomical Observatory of Japan.
- That means I am an "astronomer".
- That does not imply I have ever used any telescope to observe anything.

Then what do I study?

“Theoretical Astronomy”

In a simplified sense, its goals are:

- **to understand the observed behavior of astronomical objects in terms of knows laws of physics**
- **to extend the laws if that is really necessary**

First example of Theoretical Astronomy

- Kepler formulated, from Tycho's observations, Kepler's three laws.
- Newton showed that Kepler's laws are derived from Newtonian mechanics and Newtonian gravity.

Kepler's laws

- **The orbits of planets are ellipse with one focus at Sun.**
- $\boldsymbol{dS/dt = \mathrm{const.}}$
- $\boldsymbol{T \propto a^{3/2}}$

Newtonian equation of motion for planets: Two-body problem

$$\frac{d^2\mathbf{r}}{dt^2} = -GM\frac{\mathbf{r}}{|\mathbf{r}|^3},$$

- **Gravity from other planets were neglected.**
- **Simple closed elliptic orbits.**

$N > 2$

- Celestial mechanics: What happens if we include planet-planet interaction?
- Stellar Dynamics: How stars themselves move?

Both are very natural “next steps” from the two-body problem.

In both fields, there are significant recent advances.

Two planets

- **Simple example: Mars under the effect of Jupiter**
  - **Gravity from Mars to Jupiter is small**
- **More general case: Saturn and Jupiter**
  - **Saturn is not small**

Perturbation technique

Basic idea:

- **Start from unperturbed Kepler orbits**
- **Derive the equation for the change of orbital elements due to the gravity of Jupiter**
- **Expands it by the mass of the Jupiter**
- **Evaluate the first term (or first few terms...)**
- **(Usually assume that orbits are close to circular and close to coplanar)**

Can be extended to general cases

Success of the perturbation technique

- **Explained high-precision observations of the orbits of planets**
- **Unexplained motions led to**
  - **findings of new planets (Neptune)**
  - **Confirmation of general relativity (Mercury)**

So, is everything OK?

— not quite.

One problem:
Long-term “stability” of the solar system.

Last 20 years of stability study

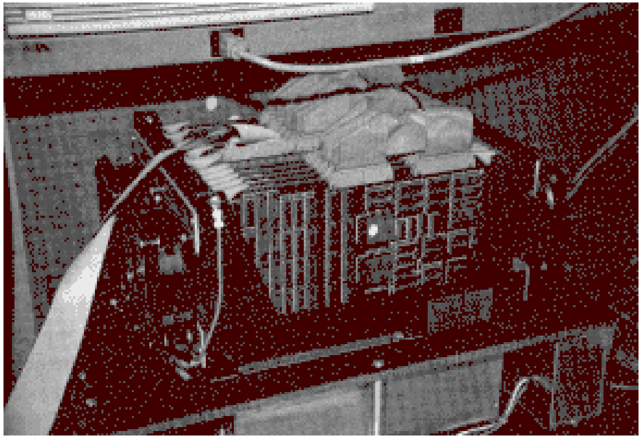
1987: Sussman and Wisdom
850Myrs numerical integration of outer five "planets"

Lyapunov timescale: 20Myrs

Lyapunov timescale: (Roughly speaking) the distance between two (infinitesimally different) systems grows in this timescale

The Digital Orrery

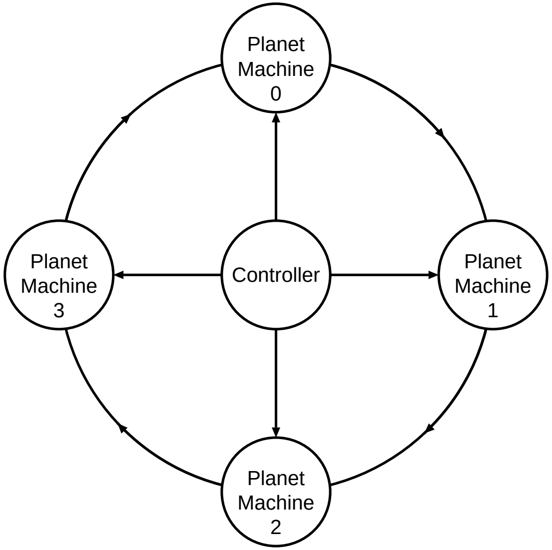
Computer used by Sussman and Wisdom



- A custom-built parallel computer for long-term integration of the Solar system
- Consists of 9 “planet computers” connected in a ring network
- 10 Mflops
- MIT AI lab + Planetary Science

Digital Orrery (2)

- **SIMD (Single-Instruction Multiple-Data) parallel computer**
- **Programmable: Integration scheme etc can be changed**
- **Effective Quadruple-precision integration**



One of (very few) examples of the successful development of special-purpose computer for numerical simulation

Naive question:

Lyapunov timescale $\ll$ Age of the solar system

Is solar system unstable? Why is it there?

Even longer numerical integration

- **Kinoshita and Nakai 1996 (4.5Gyrs)**
- **Ito and Tanikawa 2002 (45 Gyrs, 10 times the age of the solar system)**

Solar system seems to be “stable”

What do we mean by “stable”?

- Planets do not collide, exchange positions, escape from system, etc.
- not “linear stability”

Much simpler setup

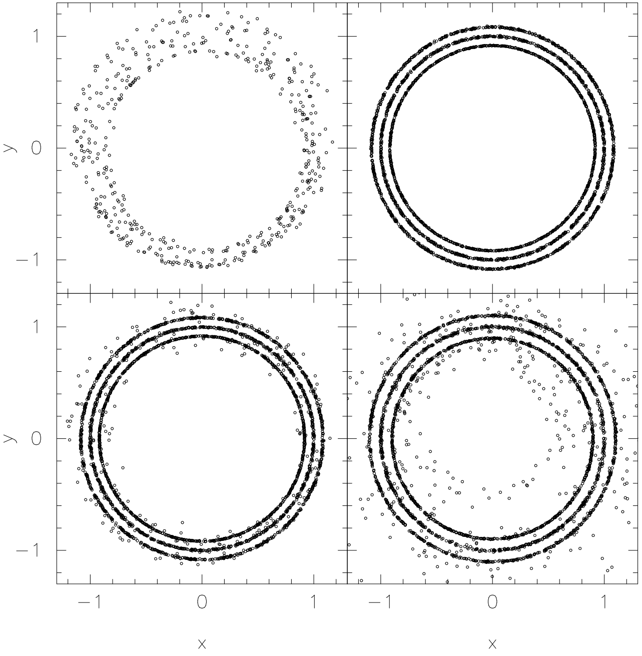
What is known:

- **Sun + two planets: STABLE (if two planets are well separated)**

What happens to the system of *three planets***?**

Simple experiment

- **planet mass: 10^{-5} (Sun=1)**
- **planet separation: 0.06, 0.08, 0.1**



Left top: 0.06, T=5000

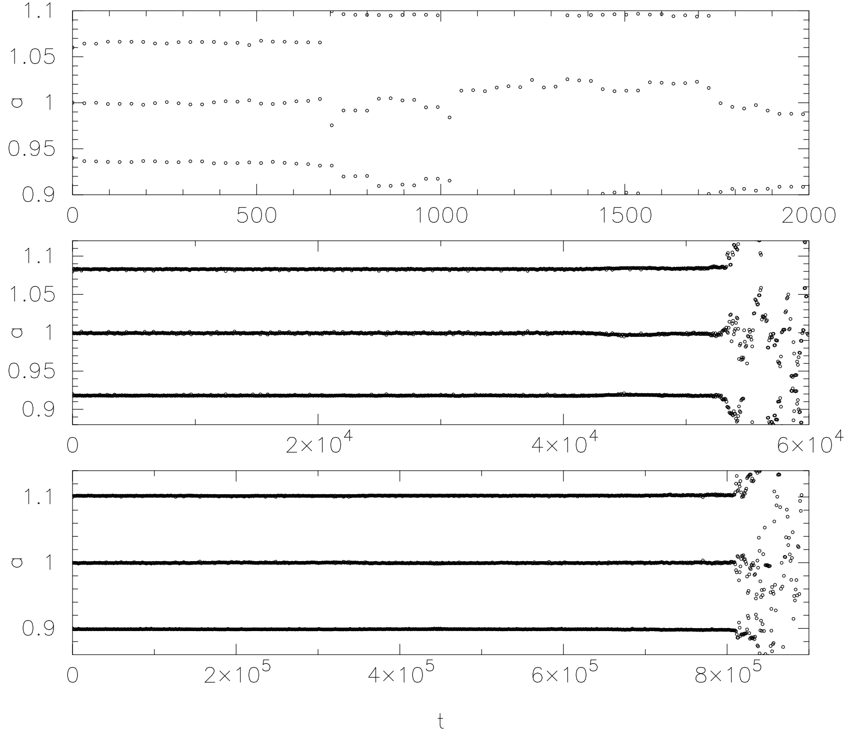
Right top: 0.08, T=50000

Left bottom: 0.08, T=60000

Right bottom: 0.1, T=90000

“Suddenly” become unstable

Evolution of semi-major axis

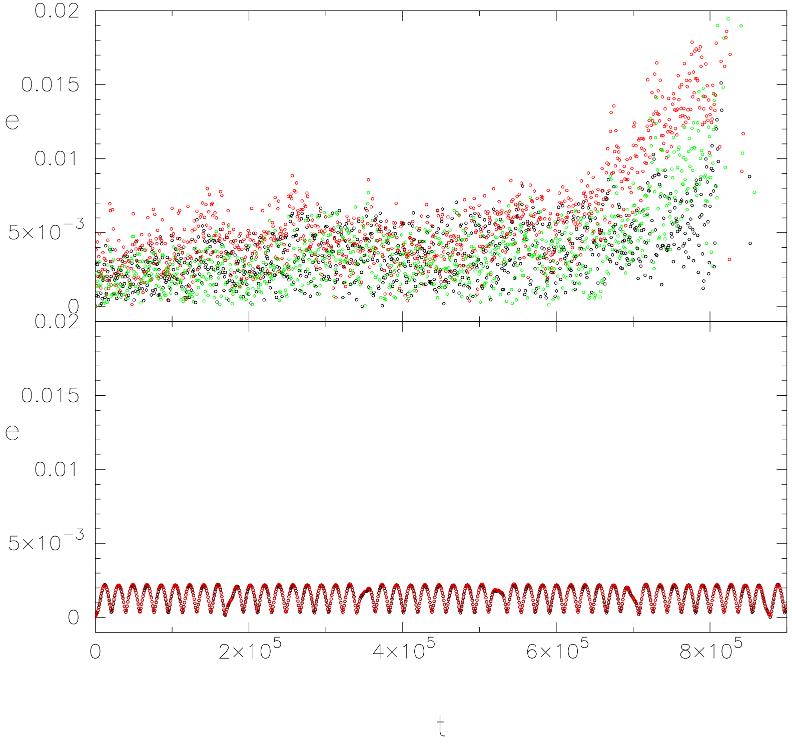


Top to bottom: separation=0.06, 0.08, 0.10

Instability timescale depends strongly on separation

Not well understood: perturbation $\propto$ separation^{-1}

Three and Two



Separation = 0.1
Eccentricities of the planets

Top: three planets
Bottom: two planets

Two planets: Stable orbits exist
Three: ???

Numerical experiments suggest:

- "Instability timescale" $\propto$ exp(separation)
- Weak dependence on the number of planets
- separation normalization: Hill radius $r_H = R(m/M)^{1/}$
- Initial eccentricity reduces the timescale

Might imply:

- Planetary system (with more than three planets) is unstable, if you wait long enough
- In the case of our solar system, instability timescale is longer than 10× its age.

Can’t we do something better than numerical integration?

Can't we do something better than numerical integration?

- **Do not ask me!**

Can't we do something better than numerical integration?

- Do not ask me!
- Ask: mathematical physicist

Is the stability of our solar system such an important problem?

— depends on whom you ask.

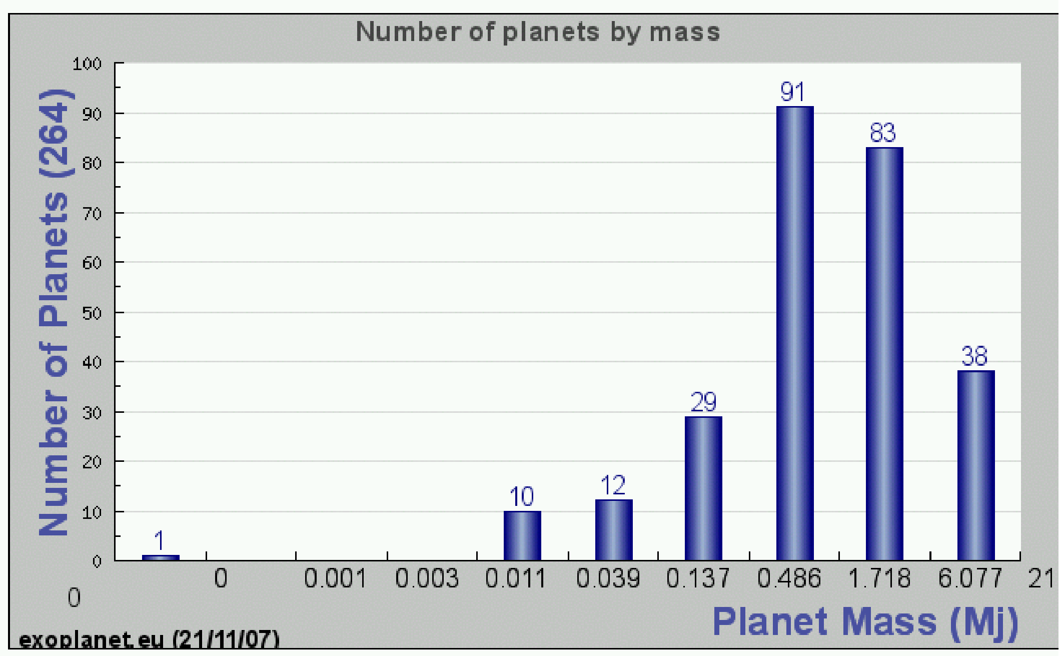
Other aspects:

- **Extrasolar planetary systems**
- **Trans-Neptunian Objects**
- **Formation theory for normal planets**

Extrasolar planetary systems

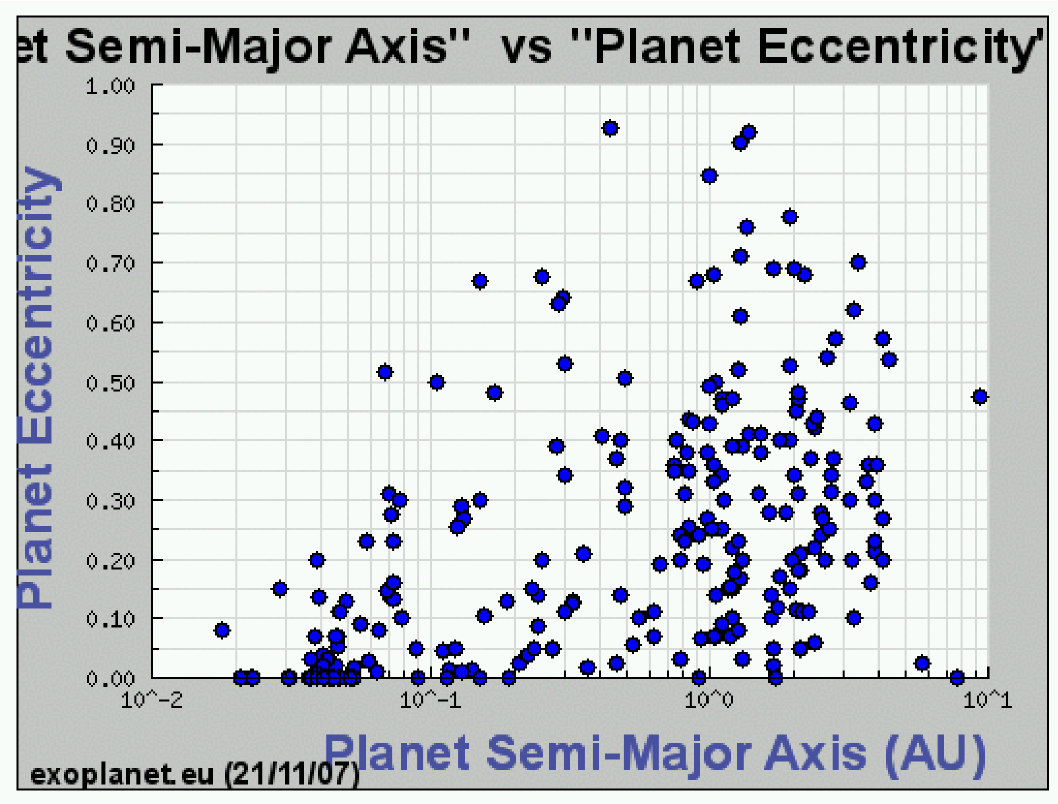
- First found in 1995 (Mayor and Queloz 1995)
- Doppler-shift measurement
  - The parent star moves around the common center-of-mass point
  - Velocity up to 100m/s
  - short-period/massive planets are easier to find

Planets mass



Typical mass $\sim$ Jupiter mass

Orbital parameters



Both a and e have wide distributions.

Limit of e: Solar tide

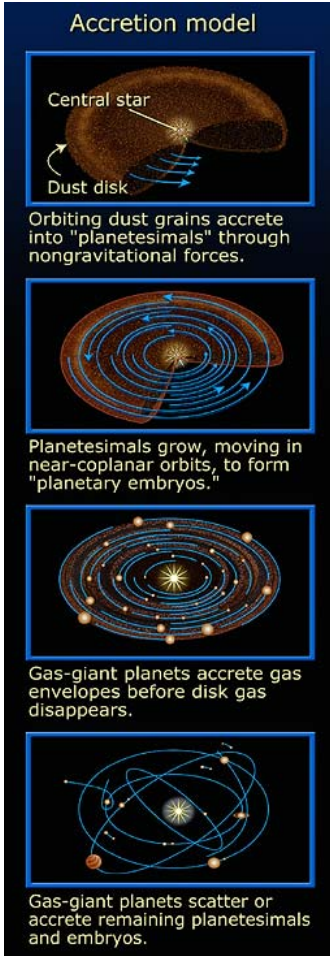
Difference from our solar system

- **Many massive planets are close to the central star (Hot Jupiters)**
- **Many planets are highly eccentric (eccentric planets)**

Theory for planet formation:

- **Massive (gas) planets can be formed outside a certain distance**
- **Planets are close to circular**

“Standard” theory for planet formation



- Planets are formed from the “dusts” in the gas disk around a star
- In the early stage, planet seeds grow through adsorption
- In late stage, they grow through gravity
- “Kyoto model”

Theoretical problems

- Gas disk rotates slightly slower than dusts
- At various stages, small seeds of planets should have fallen into the Sun because of the gas drug ("Migration")
- To make the orbits of planets close to circular is very difficult

Theoretical problems and extrasolar planets

- **“Problems” might not be so for extrasolar planets**
- **Hot Jupiters: result of migration?**
- **Eccentric planets: natural outcome of the final stage of the planet formation?**

The change in what should be explained

Before the discovery of extrasolar planets:

- **Planet formation theory should explain our solar system**
- **Might have been too narrow-minded?**

Now

- **Planet formation theory should explain wide variety of planetary systems**
- **Solar system might be rather exceptional**
- **Stability problem completely changed its meaning**

Trans-Neptunian object

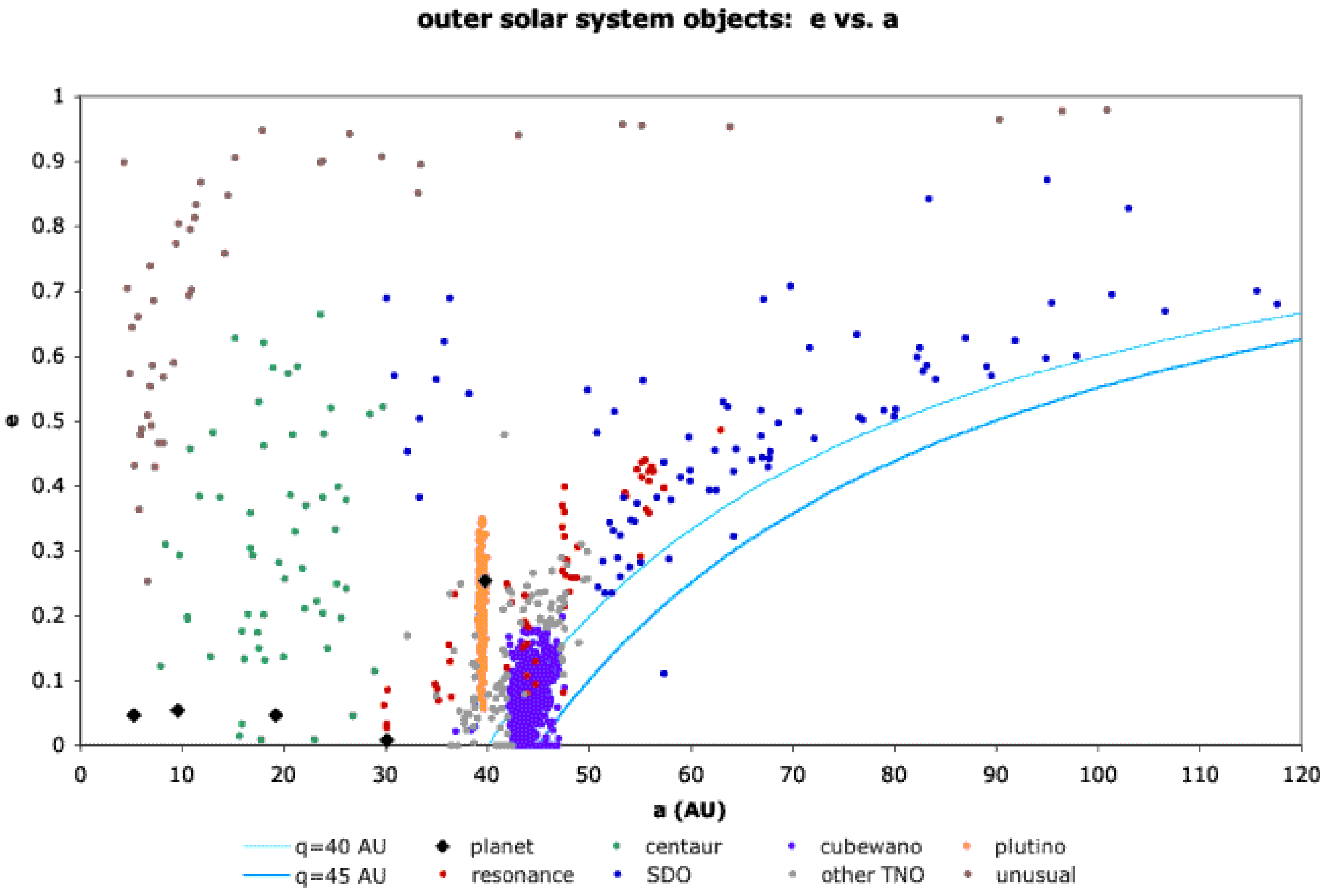
Kuiper belt:

- **Theoretical prediction that small objects should still exist outside the Neptune orbit**
- **(Standard theory cannot form Neptune anyway)**
- **There should be large number of small objects outside Pluto orbit**

Observation

- **1992: First "Kuiper-belt object", 1992QB$_1$, found**
- **More than 1,000 such objects have been found by 2007.**

Distribution of orbits



Characteristics of distribution

- **Large number of objects in 3:2 resonance with Neptune**
  - **Pluto is one of such objects**
  - **Not much in other stable resonances (in particular 2:1)**
- **Most objects lie in between 2:1 and 3:2 resonances**
- **Fair number of objects with perihelion 30-34 AU**

Current status of the theoretical understanding

- No single widely-accepted theory

One fairly successful theory: Neptune outward migration

- Neptune originally formed at 15-20AU
- Moved outward through interactions with Jupiter, Saturn, and other small objects

Successfully explained some of the observed features (not quite all)

Summary for planetary systems

N: number of planets

- **$N = 1$ solved by Newton**
- **$N = 2$: stable if large separation**
- **$N \geq 3$: Everything becomes unstable?**
  - **Why does our solar system exist?**
  - **Wide variety of extrasolar planets**

Stellar Systems

Planetary systems: Sun + “small” planets. Kepler orbit+perturbation.

Stellar systems: Consists of many stars

Examples of stellar systems

Globular clusters

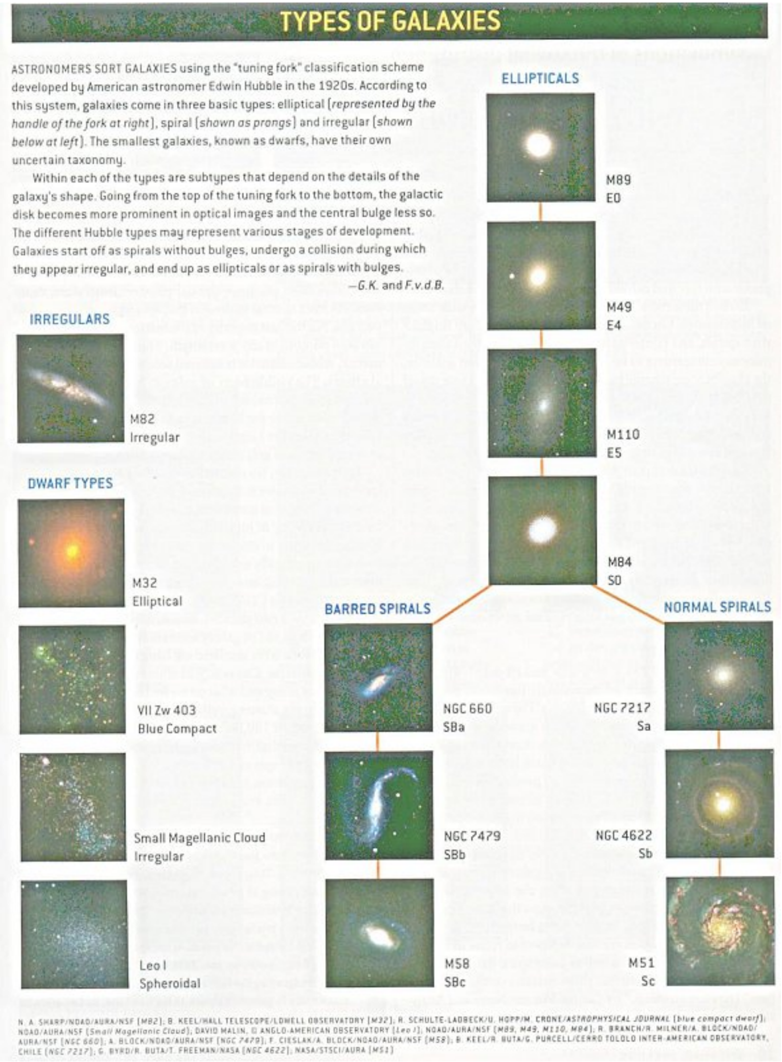
Galaxies

Globular clusters

- **10^5-10^7 stars**
- **Old stars ,$>$ 10Gyrs (age of the Universe: 13.7Gyrs)**
- **Mostly spherical (some are a bit elliptical, rotating)**
- **Globular clusters all look alike**
- **“Clean” systems, no gas, star formation etc**

Natural lab for stellar dynamics

Galaxies



TYPES OF GALAXIES

ASTRONOMERS SORT GALAXIES using the "tuning fork" classification scheme developed by American astronomer Edwin Hubble in the 1920s. According to this system, galaxies come in three basic types: elliptical (*represented by the handle of the fork at right*), spiral (*shown as prongs*) and irregular (*shown below at left*). The smallest galaxies, known as dwarfs, have their own uncertain taxonomy.

Within each of the types are subtypes that depend on the details of the galaxy's shape. Going from the top of the tuning fork to the bottom, the galactic disk becomes more prominent in optical images and the central bulge less so. The different Hubble types may represent various stages of development. Galaxies start off as spirals without bulges, undergo a collision during which they appear irregular, and end up as ellipticals or as spirals with bulges.

—*G.K.* and *F.v.d.B.*

IRREGULARS

M82
Irregular

DWARF TYPES

M32
Elliptical

VII Zw 403
Blue Compact

Small Magellanic Cloud
Irregular

Leo I
Spheroidal

ELLIPTICALS

M89
E0

M49
E4

M110
E5

M84
S0

BARRED SPIRALS

NGC 660
SBa

NGC 7479
SBb

M58
SBc

NORMAL SPIRALS

NGC 7217
Sa

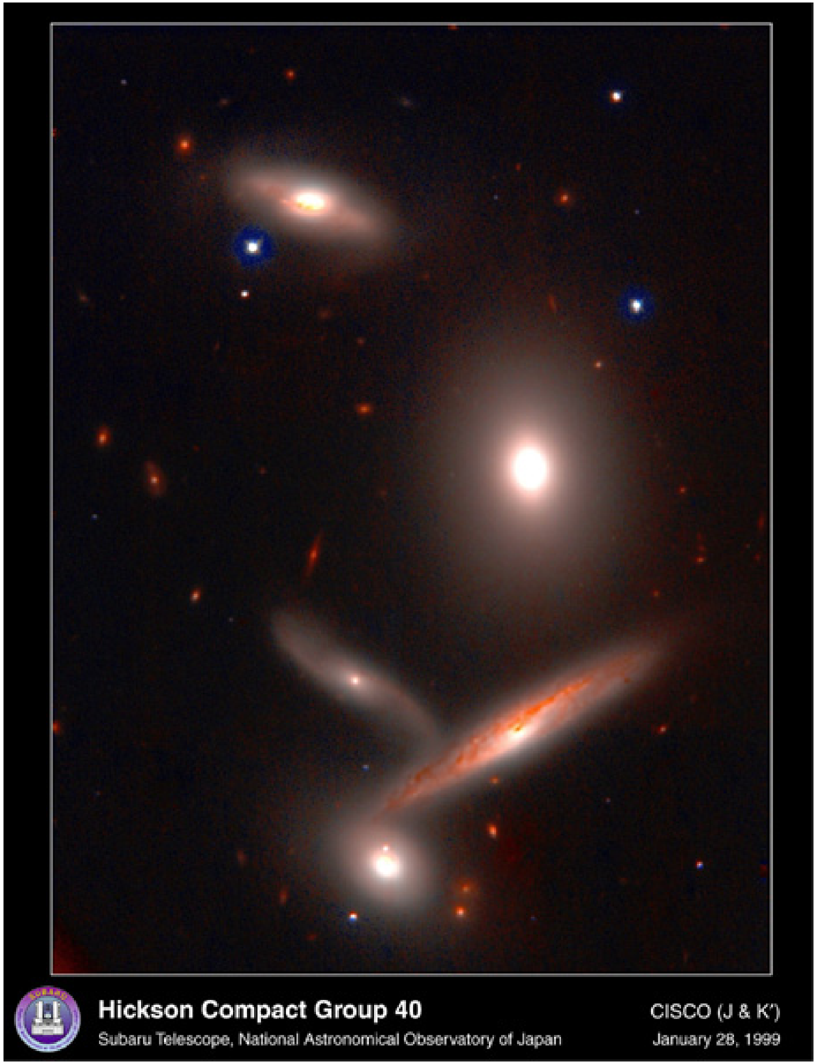
NGC 4622
Sb

M51
Sc

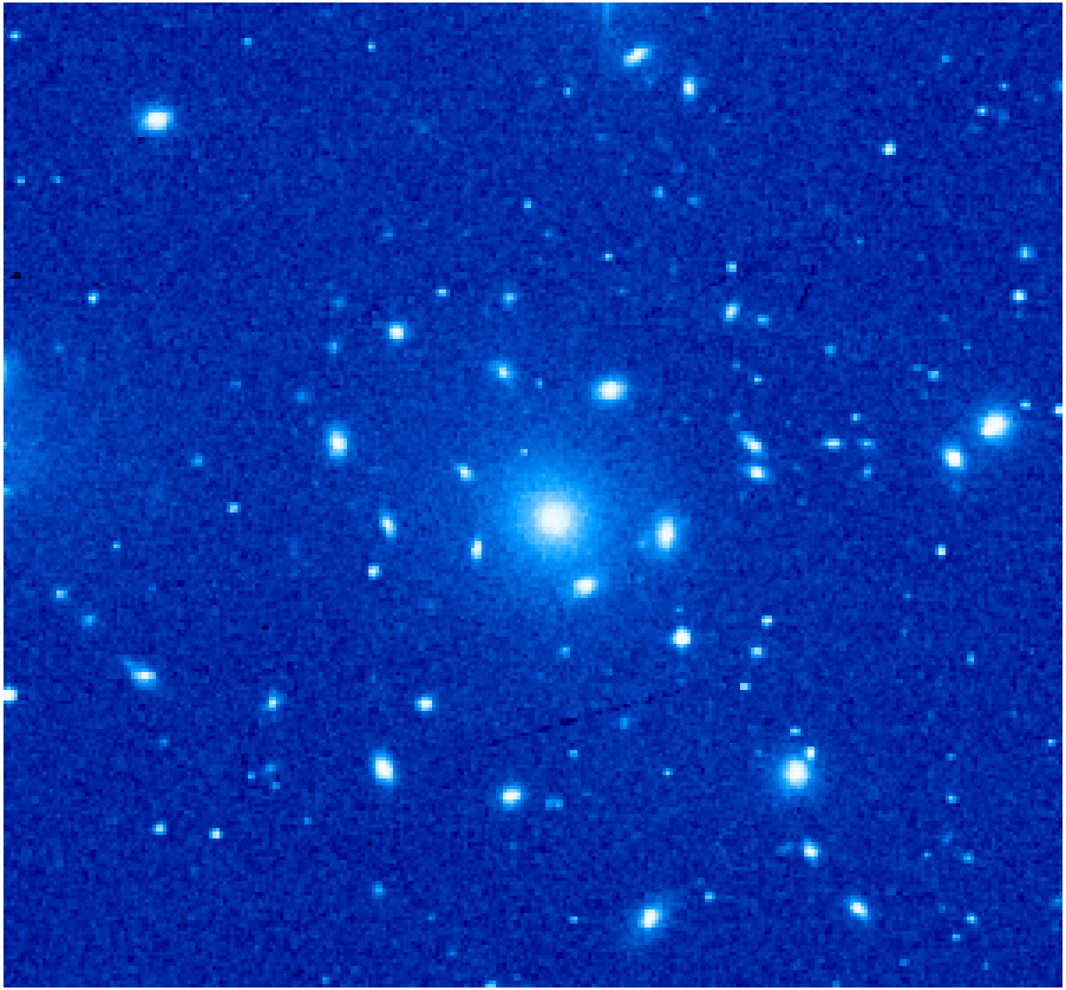
N. A. SHARP/NOAO/AURA/NSF (*M82*); B. KEEL/HALL TELESCOPE/LOWELL OBSERVATORY (*M32*); R. SCHULTE-LADBECK/U. HOPP/M. CRONE/*ASTROPHYSICAL JOURNAL* (*blue compact dwarf*); NOAO/AURA/NSF (*Small Magellanic Cloud*); DAVID MALIN, © ANGLO-AMERICAN OBSERVATORY (*Leo I*); NOAO/AURA/NSF (*M89, M49, M110, M84*); R. BRANCH/R. MILNER/A. BLOCK/NOAO/AURA/NSF (*NGC 660*); A. BLOCK/NOAO/AURA/NSF (*NGC 7479*); F. CIESLAK/A. BLOCK/NOAO/AURA/NSF (*M58*); B. KEEL/R. BUTA/G. PURCELL/CERRO TOLOLO INTER-AMERICAN OBSERVATORY, CHILE (*NGC 7217*); G. BYRD/R. BUTA/T. FREEMAN/NASA (*NGC 4622*); NASA/STSCI/AURA (*M51*)

- **$\sim 10^{11}$ stars (wide variety)**
- **Complex systems, gas, stars are forming**
- **Wide variety in morphology**

Galaxy groups

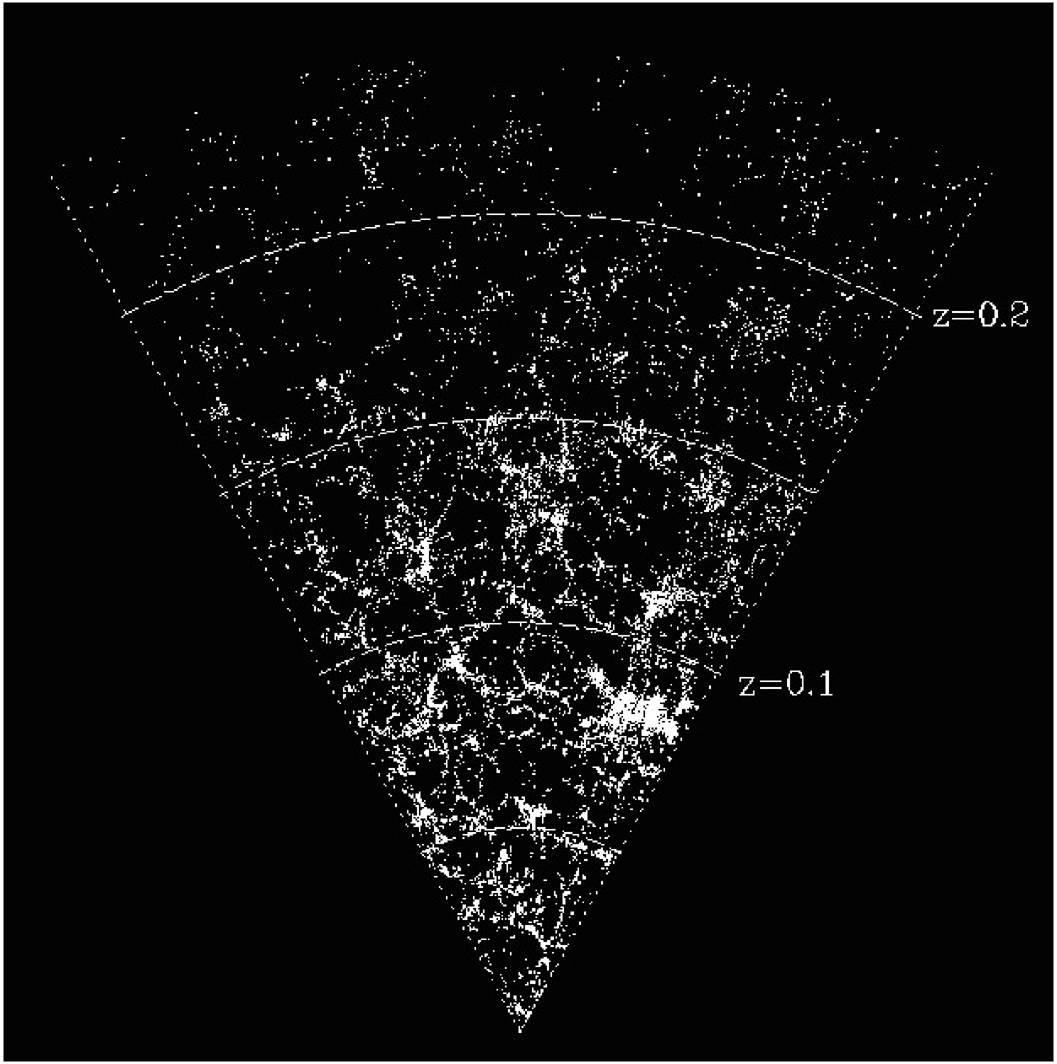


Clusters of Galaxies



http://antwrp.gsfc.nasa.gov/apod/ap950917.html

Large-Scale Structure



Old (before 1990) view of star clusters

- **Open Clusters**
  - **Pleades, Hyades**
  - **Young (Myrs - Gyrs)**
  - **Small N ($< 10^4$)**
  - **gravitationally loosely bound, or unbound**
- **Globular clusters**
  - **M5, ω Cen, 47 Tuc etc**
  - **Large $N \sim 10^6$**
  - **Old**

More resent view of star clusters

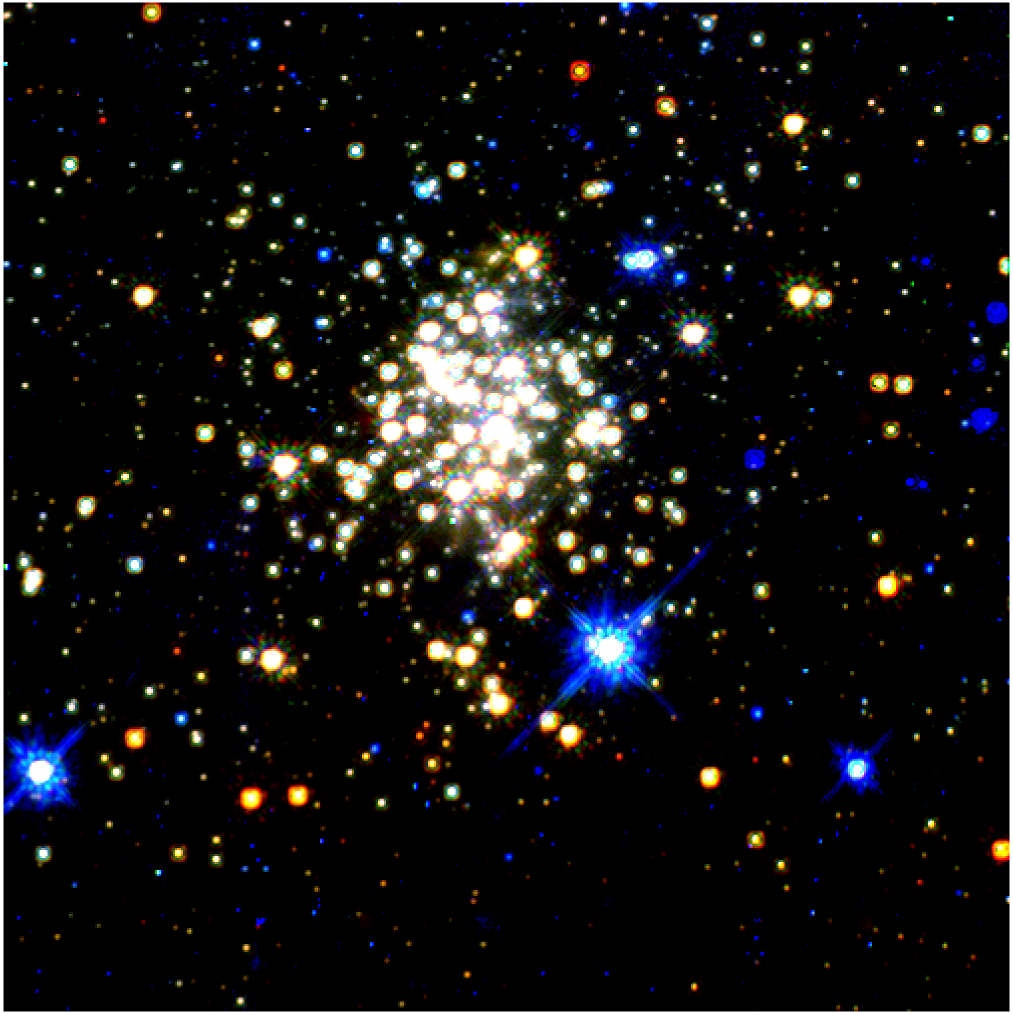
“Very young star clusters” or “Massive open clusters” have been found in:

- Magellanic clouds
- Near the center of our galaxy
- Many “starburst” galaxies

Need for new names....

More new types might be found...

Arches Cluster



About 30pc from the center of our galaxy

Mass $> 10^4 M_\odot$ (solar mass)

Radius less than 1pc

Nagata et al (1995): first found

The galactic center

As an example of very unusual star clusters

Observations in the last 15 years

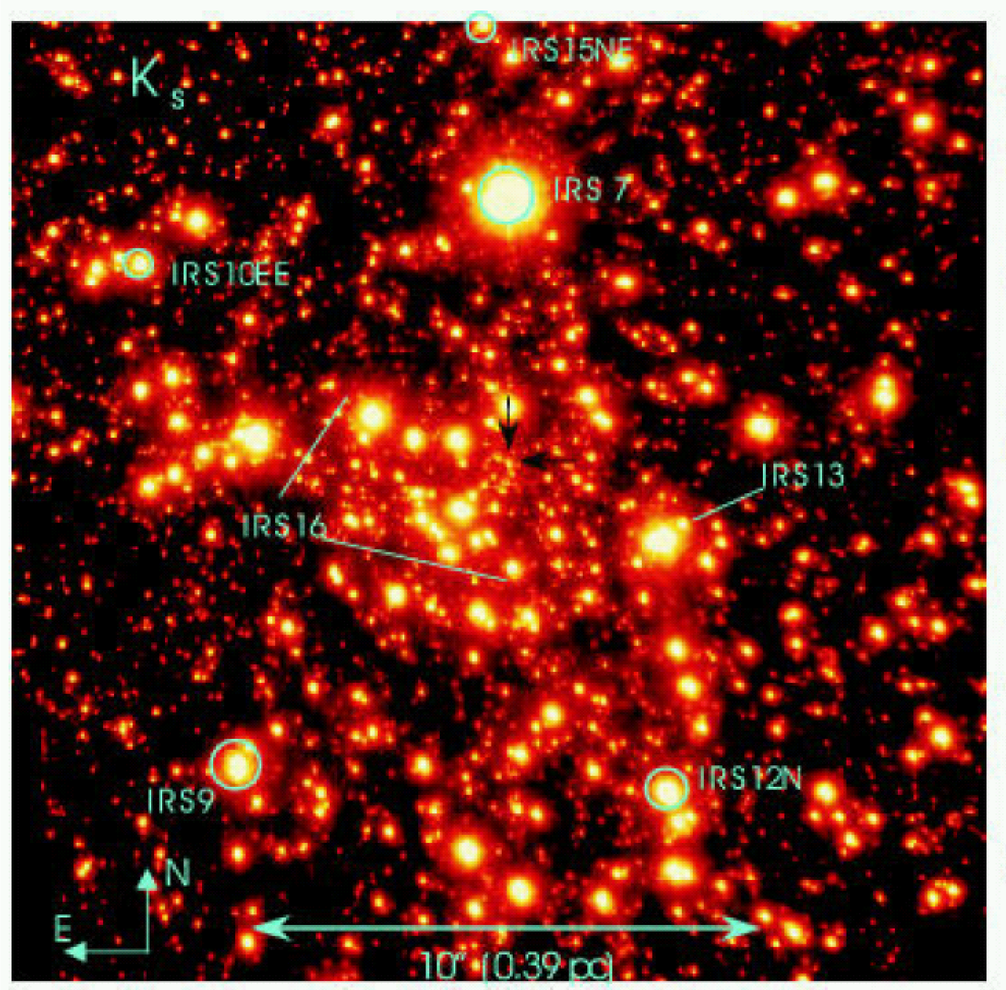
- high-resolution observation in Near Infrared, adaptive optics
- Stars which orbit around the central blackhole have been found.

The central cluster

Genzel et al 2003

K-band (2.2μm)

Black arrow points to the radio source SgrA* (the central black hole)



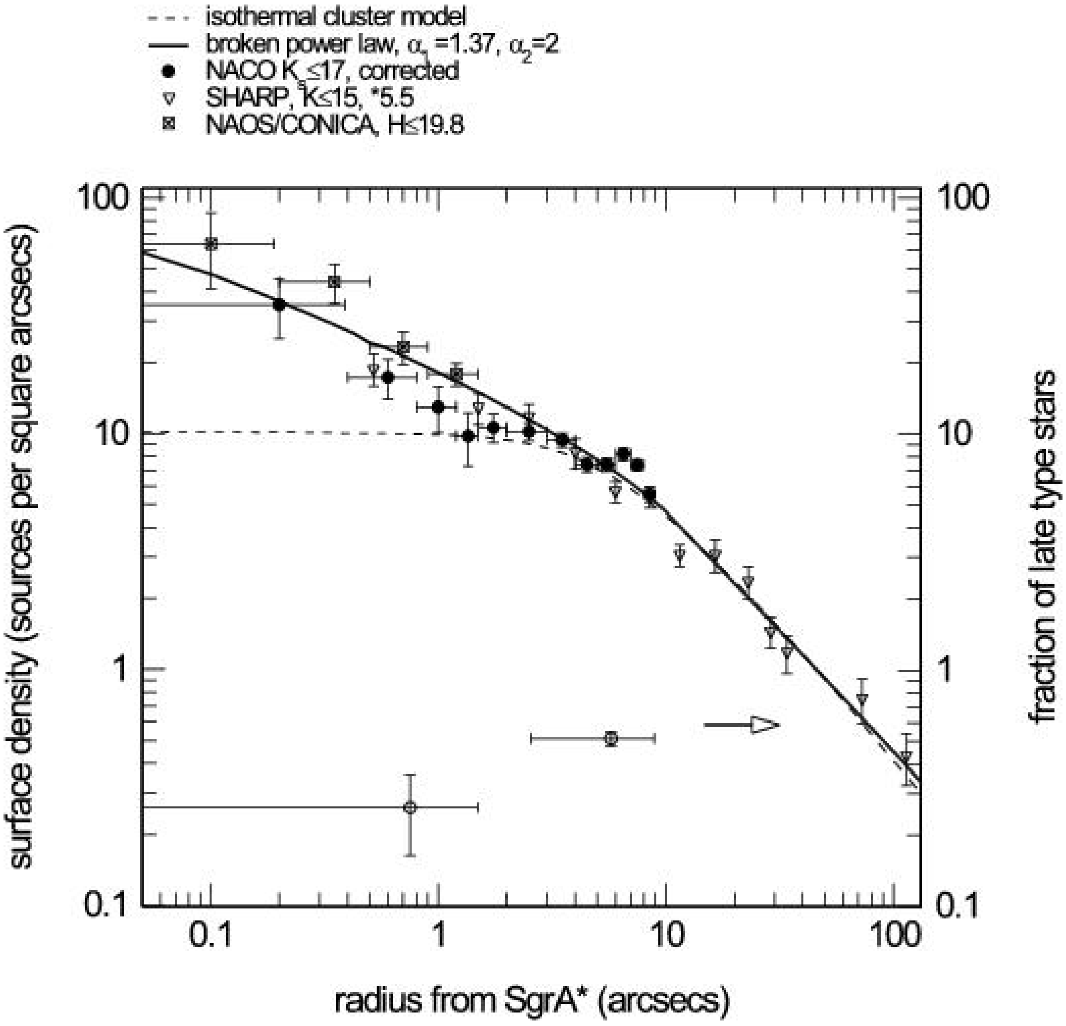
Projected number density of stars

Genzel et al 2003

Total stellar mass within 10” (0.4pc = 1.3 ly) $\sim 10^6 M_\odot$

Many young stars (A few Myrs or less)

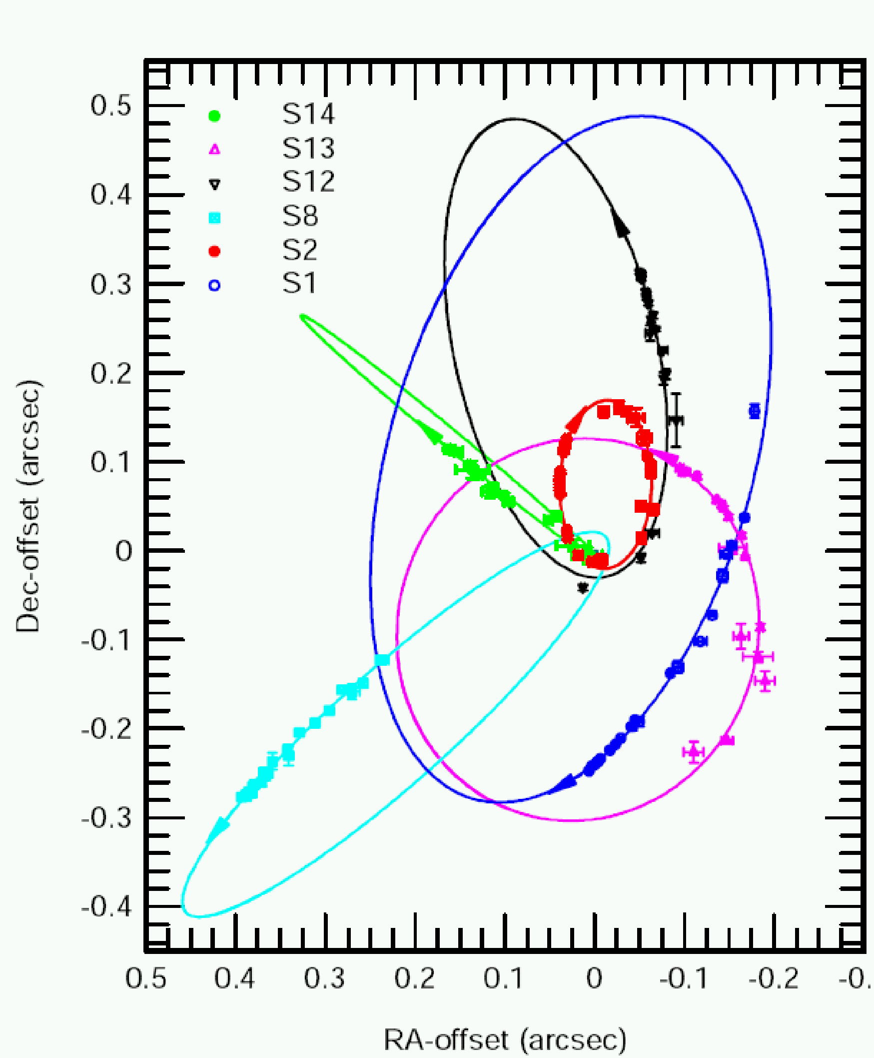
Young stars in very central region ($<$ 0.5”) (S1, S2, S0-16 ...)



Observed orbits of central stars

Eisenhauer et al 2005
In the last 15 years, motions of several bright stars with distance < 0.01pc from the central black holes have been measured.

Most of these stars are young (mass $> 10M_\odot$, lifetime $\sim$ 10Myrs)



Gravitational Many-Body problem

Equation of Motion:

$$m_i \frac{d^2 x_i}{dt^2} = \sum_{j \neq i} f_{ij} \tag{1}$$

x_i, m_i: position and mass of particle i

f_{ij}: gravitational force from particle j to particle i

$$f_{ij} = G m_i m_j \frac{x_j - x_i}{|x_j - x_i|^3}, \tag{2}$$

G: gravitational constant.

This equation, however, does not tells much about the behavior of the system.

Why not?

- the equation does not have analytic solution
- there are special cases....
  - $N = 2$
  - $N = 3$ from special initial condition
  - Solar-like systems (well....)
  - $N \rightarrow \infty$, dynamical equilibrium

On the other hand, we can numerically integrate the equation of motion using computer. Isn't that enough?

Numerical integration

In principle, numerical calculation should be enough.

In practice, it is not.

Reason:

- **Computers are not fast enough**
- **Additional physics**
    - **gas dynamics**
    - **stellar evolution**
    - **...**

Computer power and calculation cost

A naive estimate:

If we have N stars, calculation cost per timestep is N^2.

A 10^8-body system would need a computer 10^8 times faster than a 10^4-body system needs.

A more realistic estimate requires

- estimate for the cost per timestep
- estimate for the number of timestep

Evolution timescales

- **Dynamical timescale**
  - **Typical orbital period of stars**
- **Thermal timescale**
  - **The timescale in which the system approaches to the thermal equilibrium**

Thermal timescale

- **What is thermal relaxation in a stellar system?**
- **Can we apply thermodynamics?**

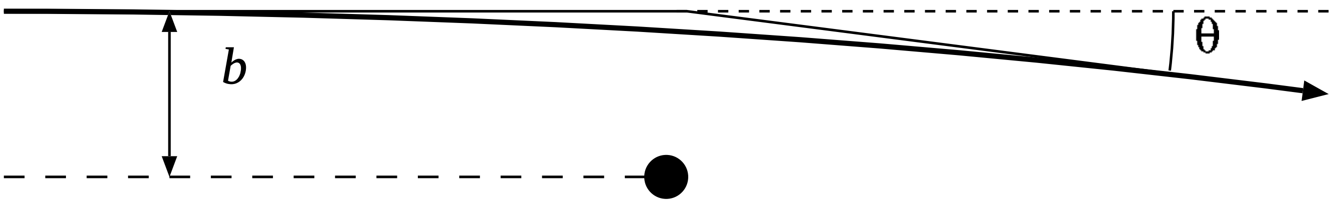
Rough estimate

Consider the following picture:

- Stars orbit under the smooth gravitational potential
- They change energy etc through close encounters with other stars

Close encounter: close enough to have large defection angle

Two-body encounter



$$\tan\theta = \frac{2b}{(b/b_0)^2 - 1}$$

$$b_0 = \frac{Gm}{v^2}$$

$M = G = 1, R \sim 1 \rightarrow v \sim 1,\ m = 1/N$

$\rightarrow b_0 \sim 1/N$

Number of close encounters per unit time

$$n\sigma v \sim N \cdot \left(\frac{1}{N}\right)^2 \cdot 1 \sim \frac{1}{N}$$

Mean collision time $\sim N$

A more accurate estimate: Thermal timescale $\sim \dfrac{N}{\log N}$

Physical meaning of thermal timescale

- **Systems with small N and/or short orbital timescale can thermally relax.**
- **Systems with large N and/or long orbital timescale cannot thermally relax.**

Typical globular clusters: $T_{th} \sim 10^9$yrs

Typical galaxy: $T_{th} \sim 10^{17}$yrs

Basic reason why globular clusters are all similar while galaxies are not.

Thermal equilibrium

Basic difficulty with statistical mechanics of stellar systems:

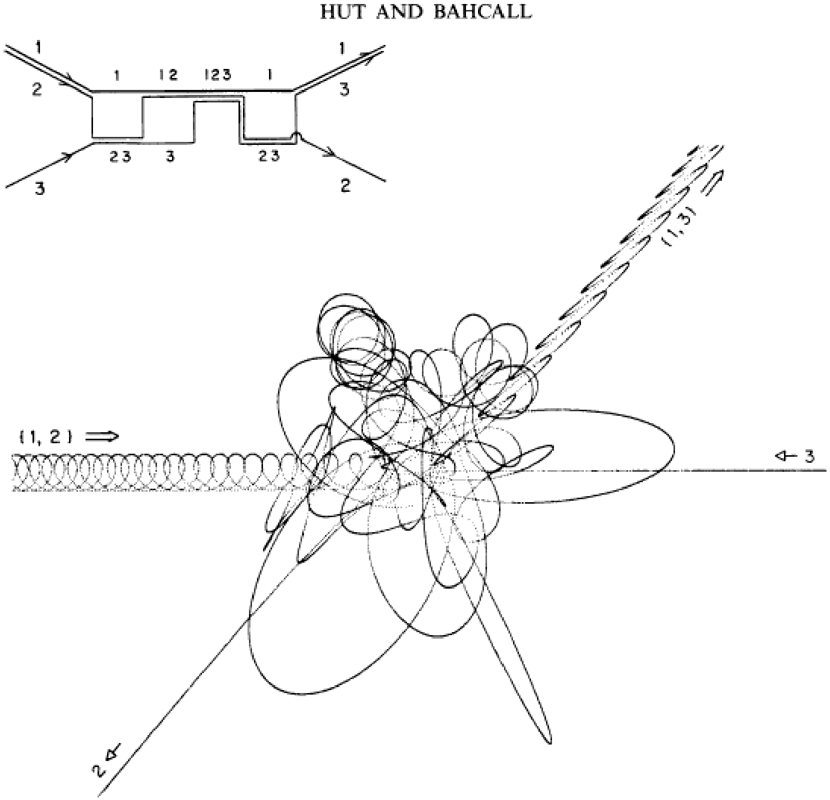
There is no thermal equilibrium.

Why?

There are many ways to demonstrate it. One example will be give here.

Simplest example — $N = 3$

Example of the numerical solution for the general three body problem:



$N = 3$ (2)

Let the system evolve from some initial condition with $E < 0$ (gravitationally bound as a whole)

Three stars orbits around each other for a while

Finally, two of the three stars become strongly bound and the third one is ejected out of the system.

This is the "final state" for three-body problem

Essentially the same for large N...

Study of stellar system before the final state

Numerical integration is useful

In many cases computers are still not fast enough

- Improve the numerical method
- Buy fast computers
- Build fast computers

Numerical methods

Basic need:

Numerically integrate the equation of motion:

$$\frac{d^2\boldsymbol{x}_i}{dt^2} = \sum_{j\neq i} G m_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^3}, \tag{3}$$

Program to calculate the right-hand side:

double loop, 10 lines or so?

Time integration: Just use some reasonably accurate scheme, *e.g.*, **Runge-Kutta?**

Unfortunately, things are not so simple....

What is the trouble?

- Accuracy problem: close encounters between two particles, structure formation requires very short timesteps, while the overall integration time is very long [$O(N)$].
- Calculation cost: Right-hand side is $O(N^2)$, simulation time adds another N — large-N systems become too costly (cannot finish even with fastest computers).

Calculation methods in time domain and space domain.

Time domain

Essentially just an initial-value problem for the system of ordinary differential equation.

One might think we can just use some well-known scheme in some mathematical software library.

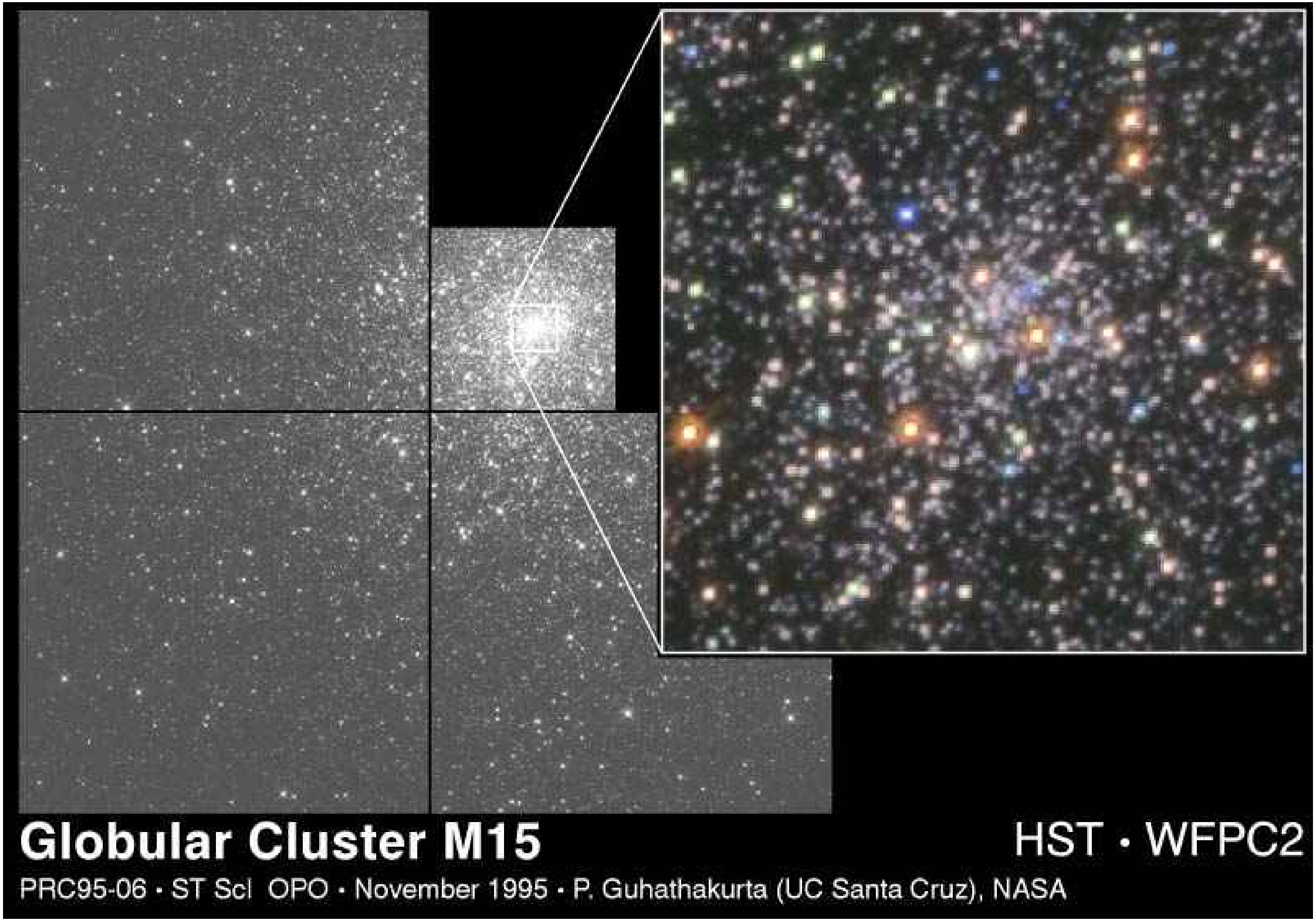
We cannot, because:

- **Different stars can have very different timescales**
- **Binary stars and other small-N subsystems require special care.**

Timescale problem

- **Effect of structure formation**
- **Problems which arise even in “uniform” systems**

Effect of structure formation



Globular cluster M15

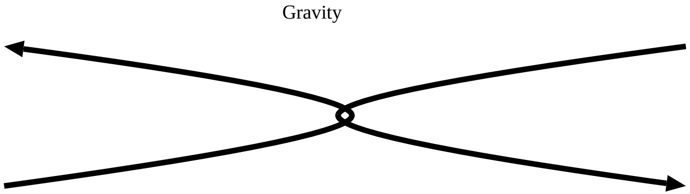
- **“Core collapsed” cluster: Number density of observed stars rises as $r^{-1.8}$**
- **Central massive black hole?**
- **central clump of “invisible” stars? (Neutron stars, white dwarfs)**

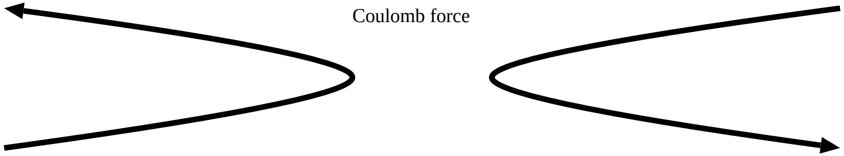
Problems which arise even in “uniform” systems

Since the gravity is pure attractive force, two stars can approach arbitrary close

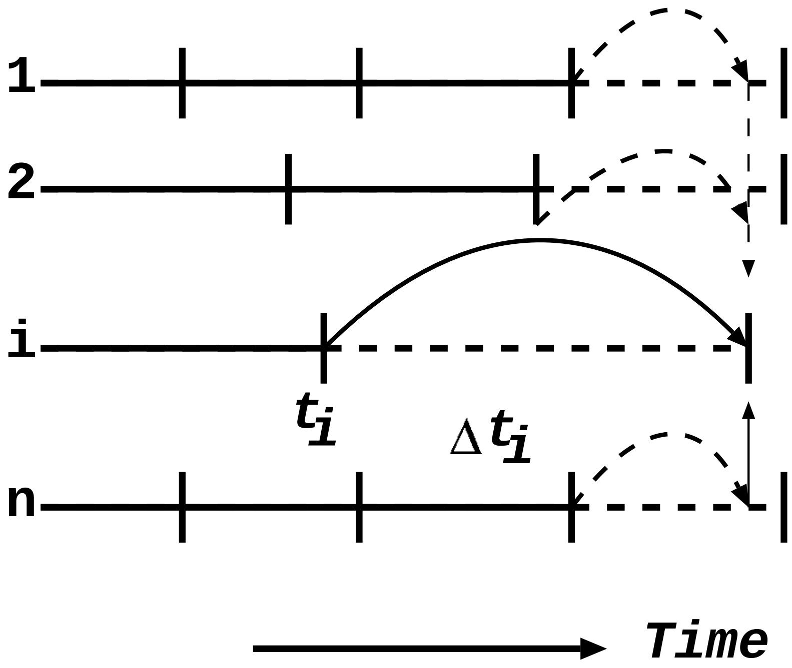
Very small timesteps become necessary

Unique problem of gravitational many-body systems. Molecular dynamics does not suffer this.





Individual timestep



(Aarseth 1963)

- Each star has its own time and timestep
- Event-driven integration: — star with minimum $t_i + \Delta t_i$ is selected

Requirements for integration scheme

- **High-accuracy predictor necessary**
- **Variable stepsize necessary**
- **Cannot use scheme which require the calculation of acceleration at intermediate points (eg: Runge-Kutta)**
  - **Linear Multistep method OK**
  - **Runge-Kutta not OK**
  - **Symplectic schemes not OK**

Space domain

How do we calculate the right-hand side of the equation of motion?

For a while we forget about the individual timestep scheme...

Widely used method: Barnes-Hut treecode

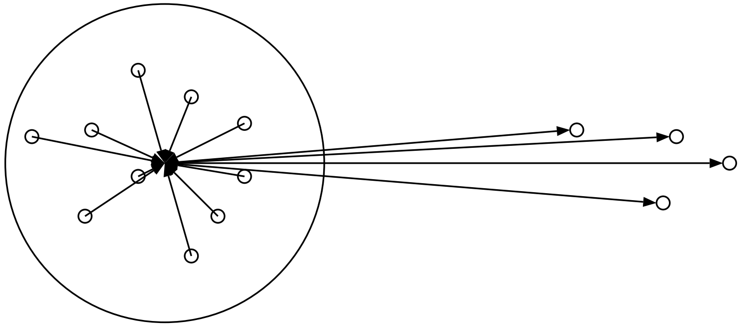
Widely know method: Fast-multipole method (FMM)

Basic idea for tree method and FMM

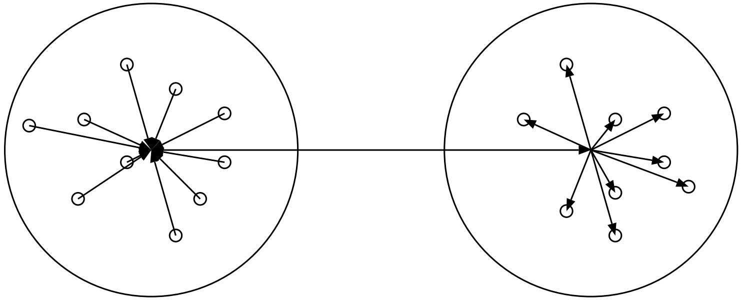
Force from distant particle: Weak

↓

Can't we evaluate many forces at once?



Tree



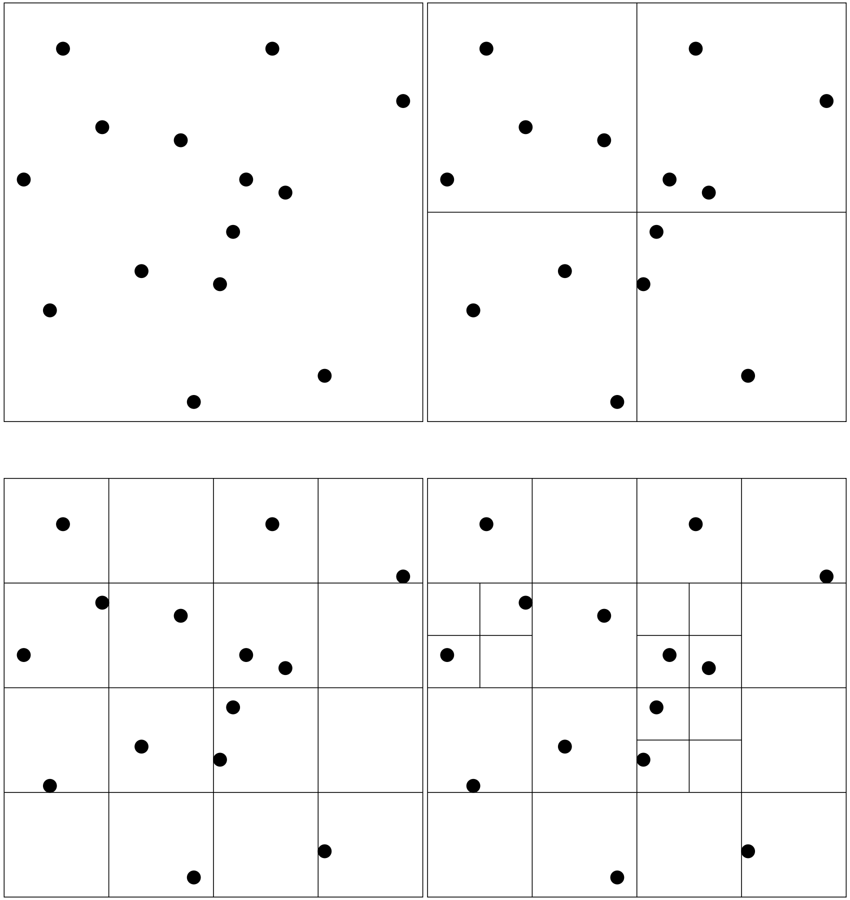
FMM

- **Tree: aggregate stars which exert the forces**
- **FMM: aggregate both side**

How do we aggregate — Barnes-Hut tree

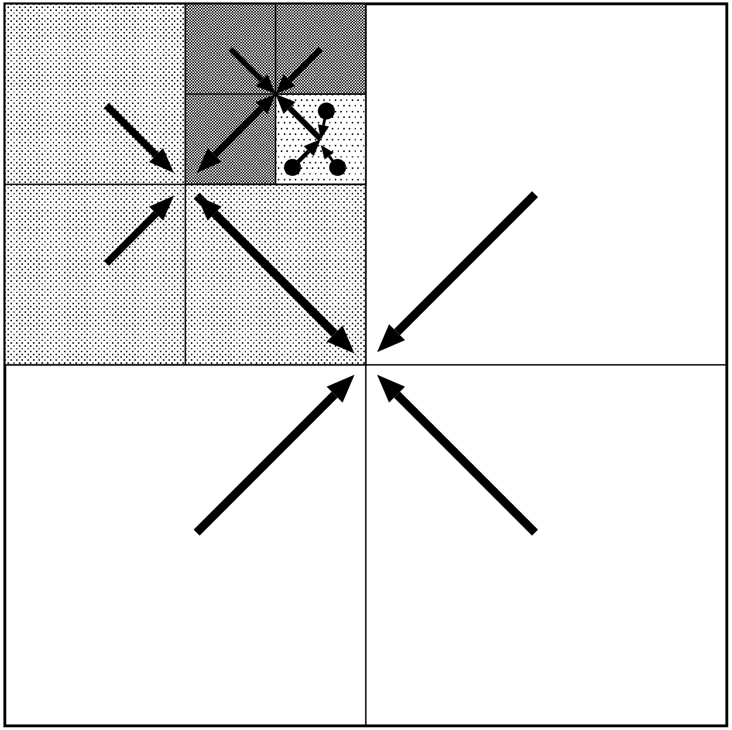
Use tree structure

- First make a cell with all stars in it
- Recursively subdivide the cells to 8 subcells
- Stop if there is small enough stars



Construction of the multipole expansion

Form the expansion for cells.

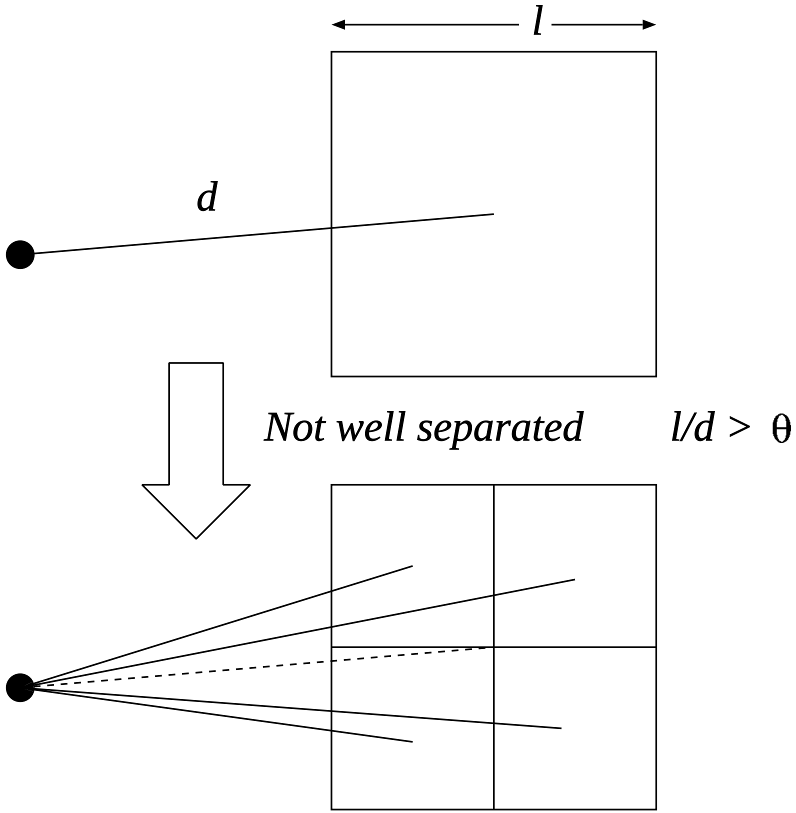


- **lowest-level cells: Directly calculate the expansions for stars in it.**
- **Higher-level cells: Shift and add the expansions for child cells.**

Calculate bottom-up.
Calculation cost: $O(Np^4)$ (p: expansion order)

Force calculation in tree method

Recursive expression:



- **Well separated: apply the multipole expansion**
- **not: take summation of the forces from the child cells**

Total force = force from the root cell

Second approach: Use fast computer

We can do fast calculation by using fast computer.

... not that simple ...

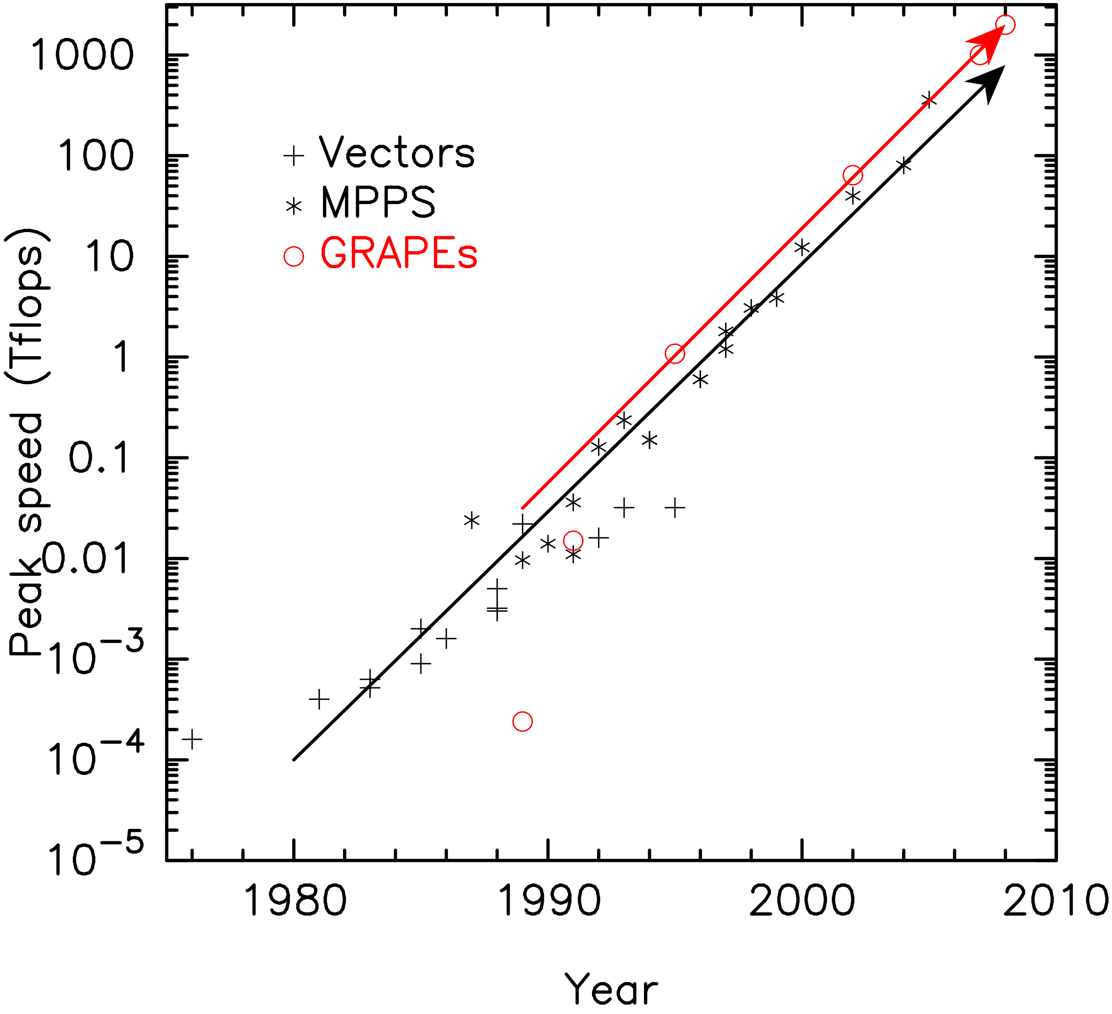
Basic reason:

The development of high-performance computers in the last 30 years made it more and more difficult to use them

Advance in computers

Speed improvement: 10^{10} in 50 years

Roughly exponential in time



How the exponential increase made possible?

1. **Moore's law: Size of transistors halves every three years**
    - **4 times more transistors**
    - **2 times faster**
2. **Change in computer architecture**
   Scalar → Vector → distributed parallel

We need parallel algorithm which is efficient on parallel machines with relatively slow network

(I'll not discuss it here...)

Third approach — build your own computer

Using fast computers is not easy...

- In 10 years, computer architecture might completely change, making your program totally useless.
- Using modern machines is hard:
  - Parallelization on distributed-memory machine
  - Cache reuse
  - Other complicated techniques

Isn't there a somewhat better way of life?

One approach: build your own computer

It's difficult to use the computer somebody else mad for some other purpose

Could be simpler to design the machine suited for your goal (special-purpose hardware).

Why consider special-purpose?

(Might be) faster and cheaper than general-purpose computers.

Why?

- **Characteristics of the problem itself**
- **Technical aspects**
- **Historical, economical aspects**

Characteristics of the problem itself

Stellar system :one star interacts with all other stars

- Large calculation cost (compared to memory requirement)
- Calculation is simple loop
- Communication pattern is simple

We do need some additional considerations for individual timestep and tree code.

Classification of the physical systems

Continuous:(Hydro etc): regular, near-neighbor communication, small calculation cost

Particles: regular $N \times N$ comm), high calculation cost

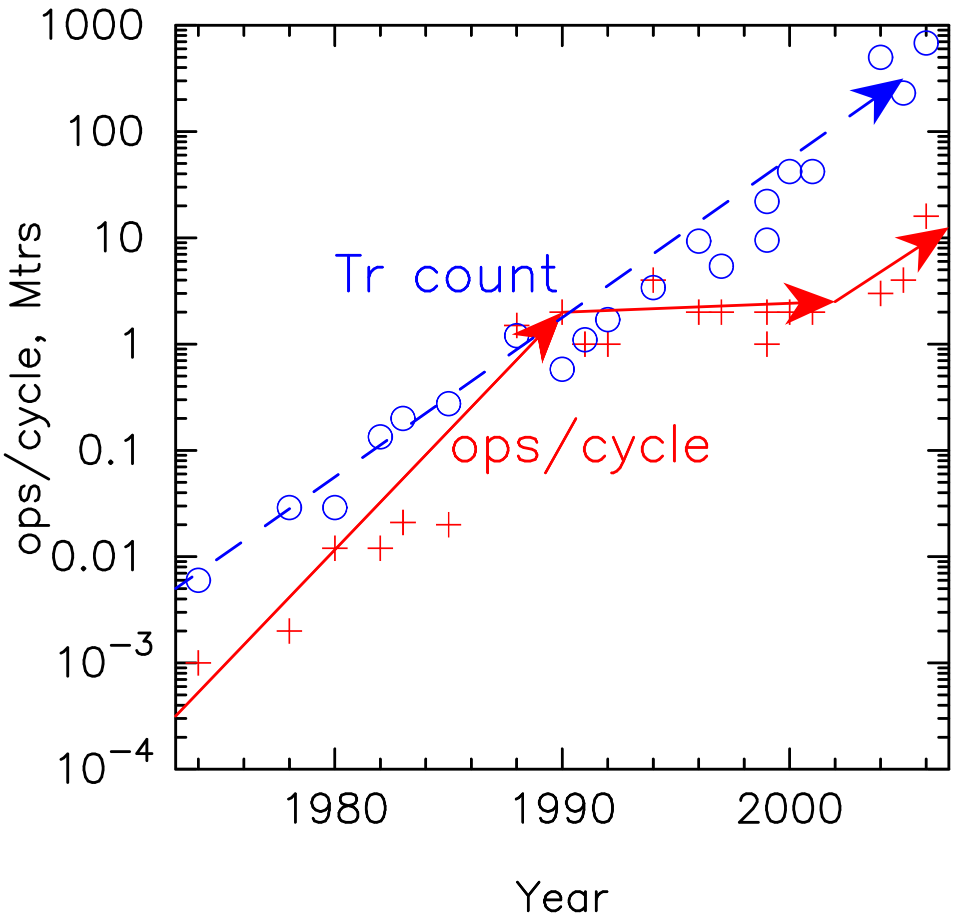
Others(discrete irregular systems)

Regular and costly = suited to special-purpose hardware

Technical aspects

- **Advance in semiconductor technology: Large-scale circuits with large number of arithmetic units becomes technologically feasible**
- **Limit in design method = rapid decrease in transistor efficiency**

“Evolution” of microprocessors

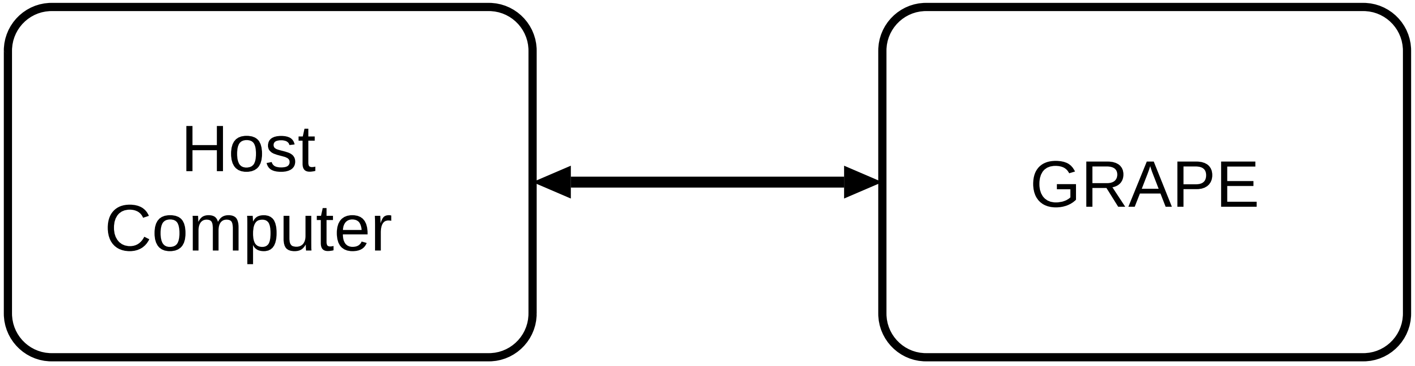


Number of transistors and Number of arithmetic operations per clock cycle

Transistor number increases exponentially
Operation count stuck at 1

Could be improved?

Basic idea of GRAPE



Time integration etc.

Interaction calculation

Special hardware:	**interaction calculation**
General-purpose host:	**everything else**

Special-purpose hardware

- Pipeline processor specialized for interaction calculation
  - Large number of FPUs
  - Small overhead
  - All FPUs always run in parallel

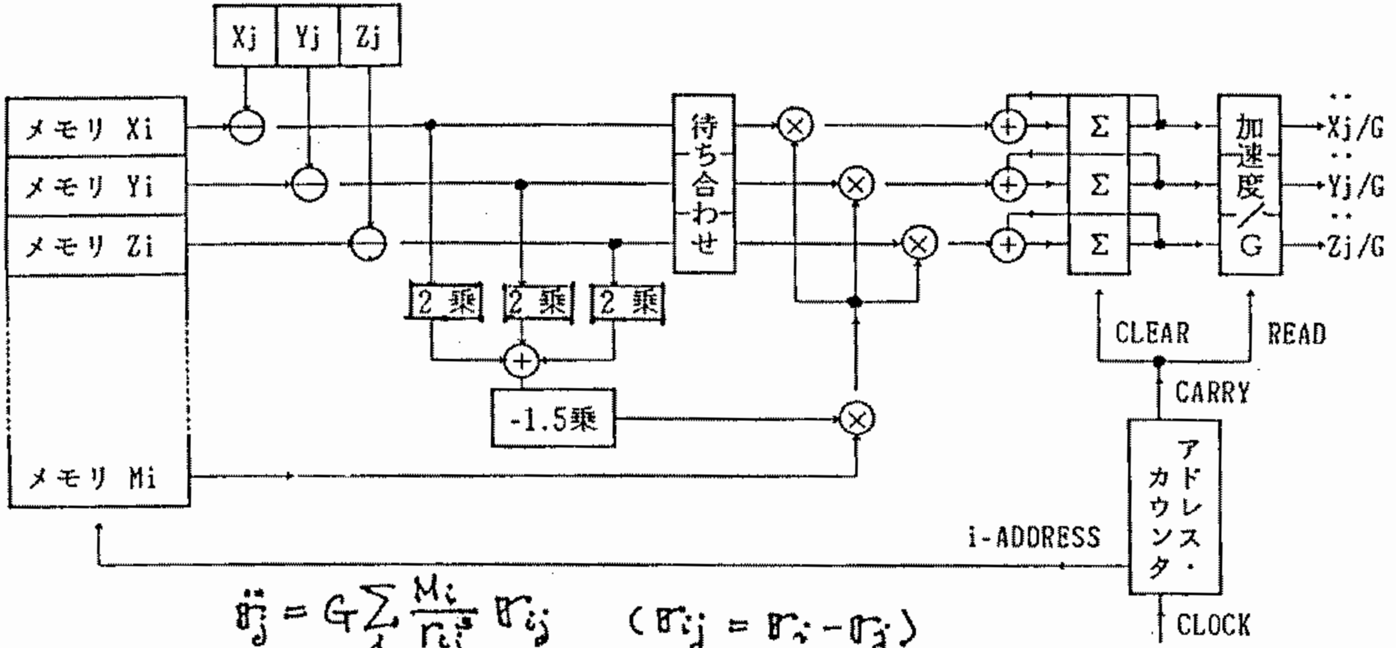
  → Very high performance

Important condition: low memory bandwidth requirement

General-purpose host computer

- “High-level” languages (Fortran, C, C++...)
- Existing programs with minimal changes
- Individual timestep, tree method, FMM

GRAPE Pipeline



$$\ddot{\mathbf{r}}_j = G\sum_i \frac{M_i}{r_{ij}^3}\mathbf{r}_{ij} \qquad (\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j)$$

+, −, ×, 2乗は1 operation, -1.5乗は多項式近似でやるとして10operation 位に相当する.
総計24operation.
各operation の後にはレジスタがあって, 全体がpipelineになっているものとする.
「待ち合わせ」は2乗してMと掛け算する間の時間ズレを補正するためのFIFO(First-In First-Out memory).
「Σ」は足し込み用のレジスタ. N回足した後結果を右のレジスタに転送する.

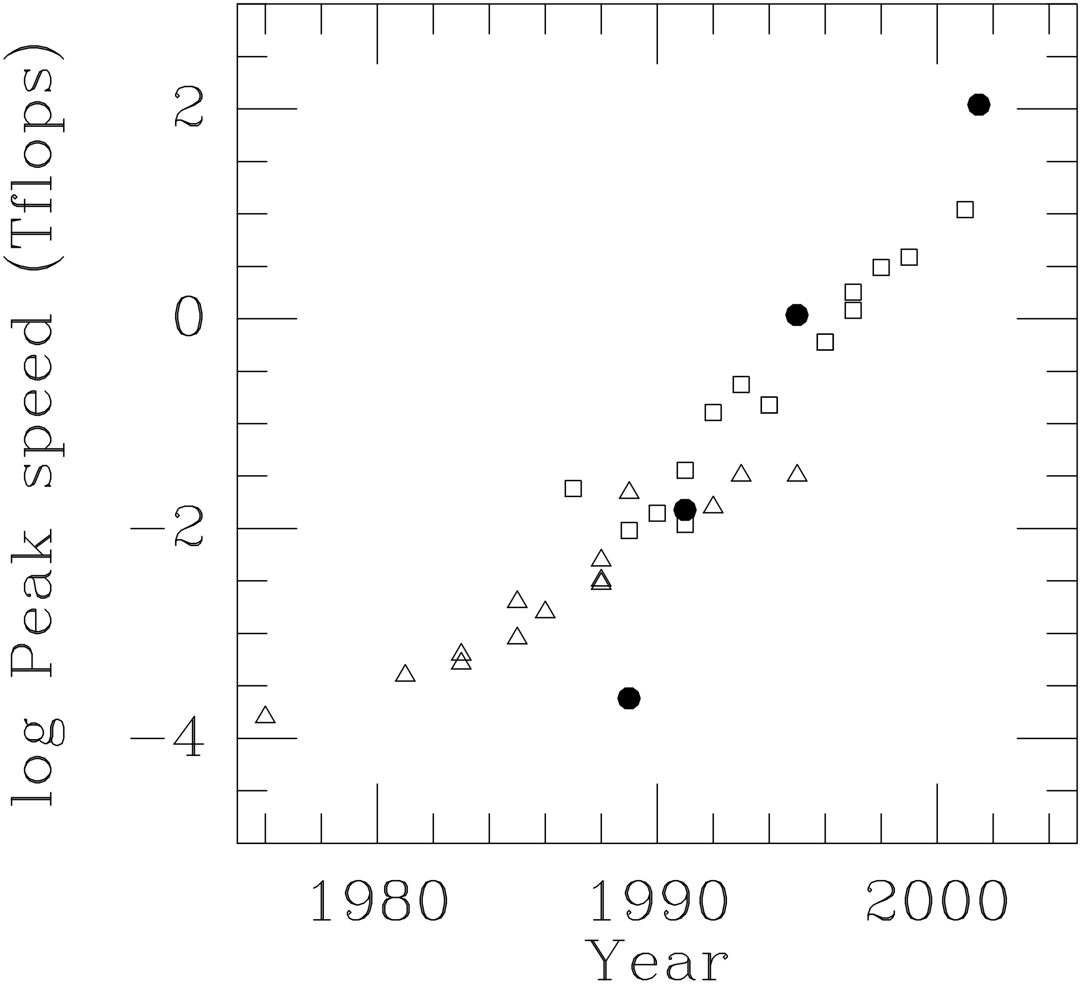
図2. N体問題のj-体に働く重力加速度を計算する回路の概念図.

(Chikada1988)

Evolution of GRAPEs

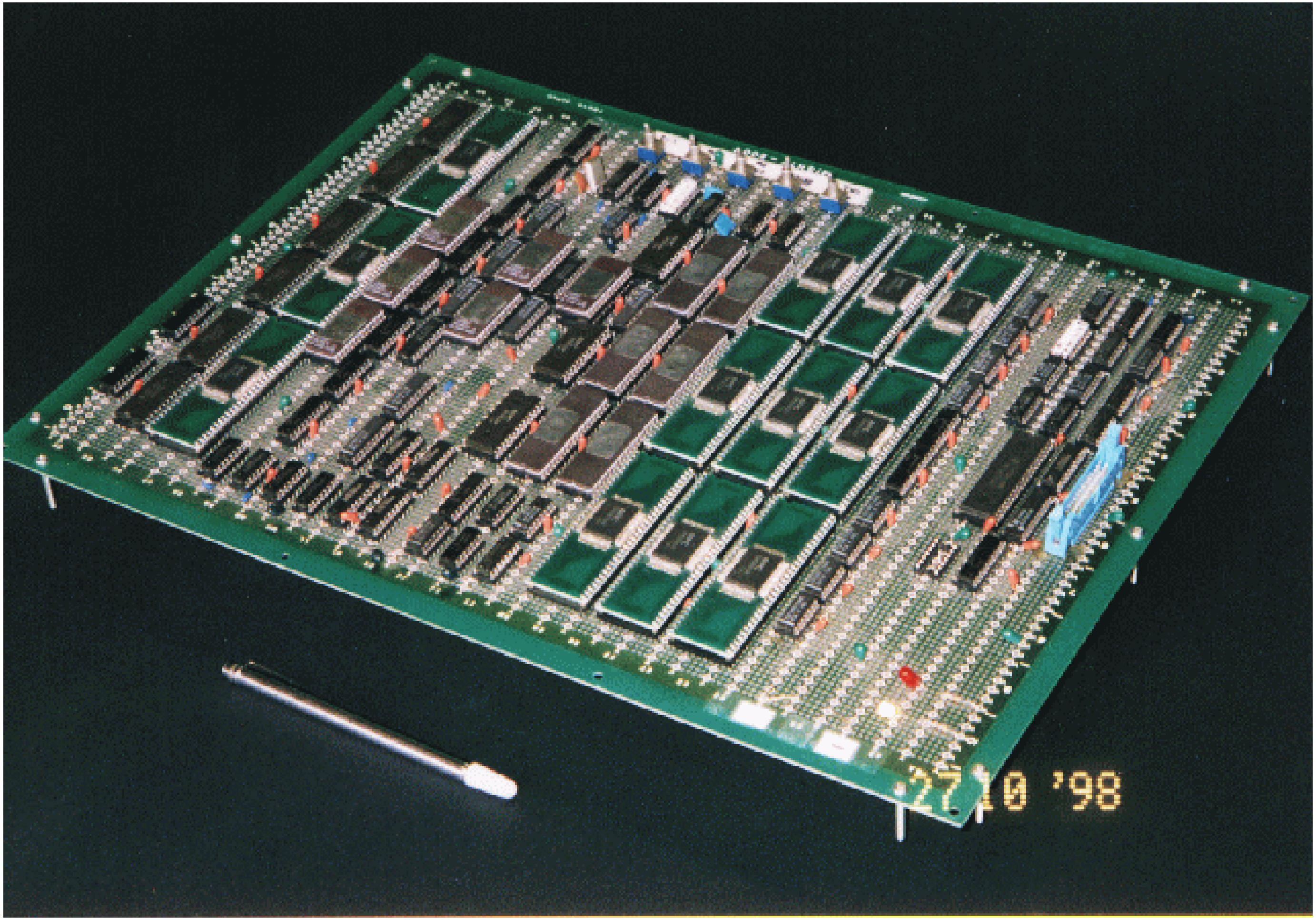
1989	**GRAPE-1**	**low acc, EPROM**
1990	**GRAPE-2**	**high acc, FPU chips**
1991	**GRAPE-3**	**low acc, Custom LSI**
1995	**GRAPE-4**	**high acc, custom LSI, Massively Paral**
1998	**GRAPE-5**	**low acc, two pipes in a chip**
2001	**GRAPE-6**	**high acc,six pipes in a chip, MP**
2005	**GRAPE-7**	**low acc, 20 pipes in a chip**

Evolution of speed



filled circles: GRAPE

GRAPE-1



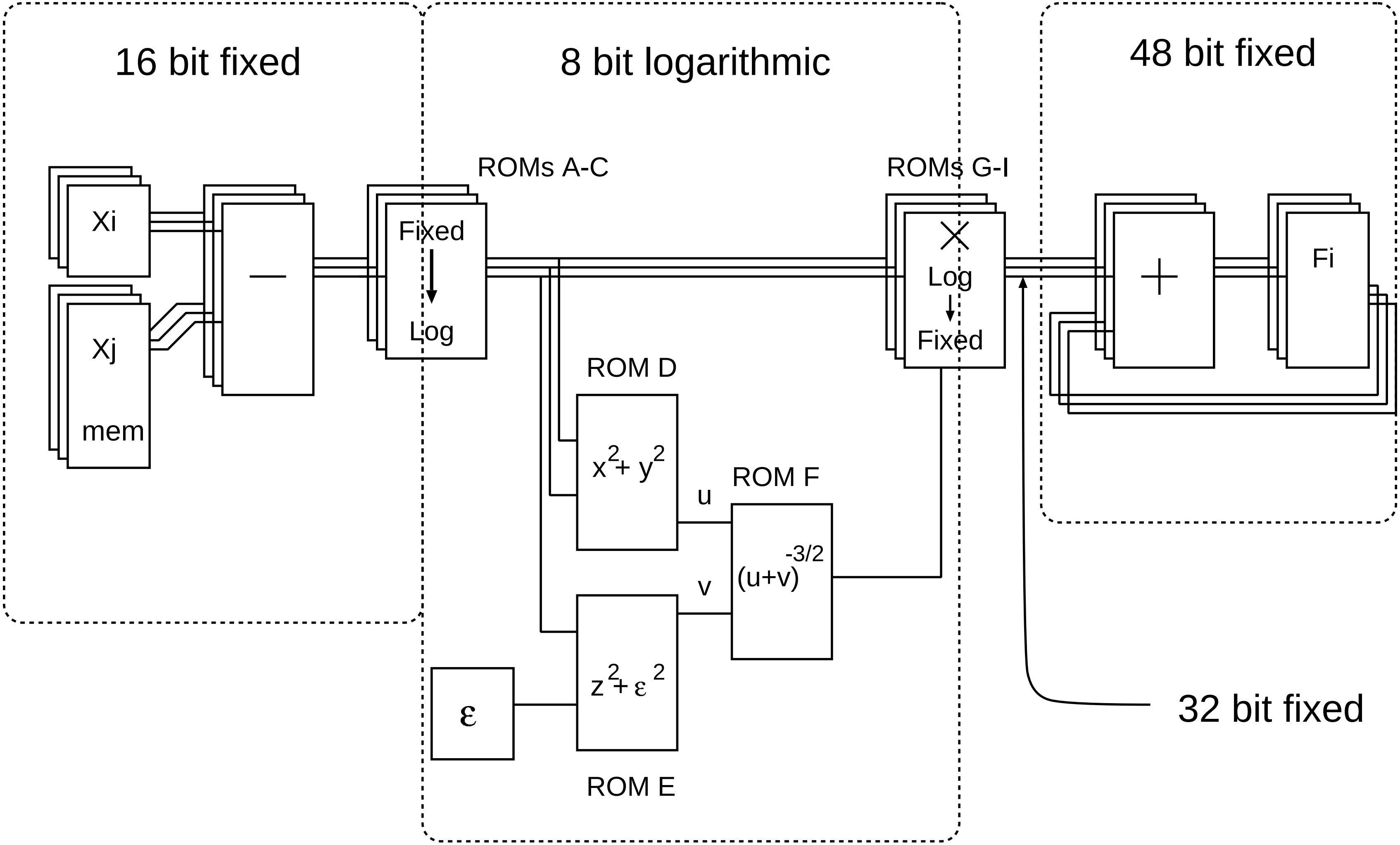
GRAPE-1 internals

“Digital Circuit for the beginners”

Initial goal

- Make something like a force pipeline
- Connect to the host and evaluate performance
- Do not care much if it is useful for real calculation

GRAPE-1 pipeline



Troubles during development

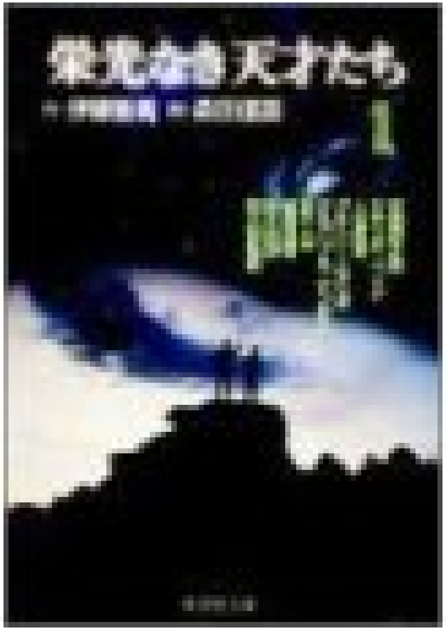
Hardware seemed to be completed without much problem (since Ito did the work, not me)

Performance problem:

Initially we used one MS-DOS PC (NEC PC-98). It was fine

We moved to a Unix workstation (Sony NEWS): Communication became very slow
We had to hack the operating system...

Tomoyoshi Ito

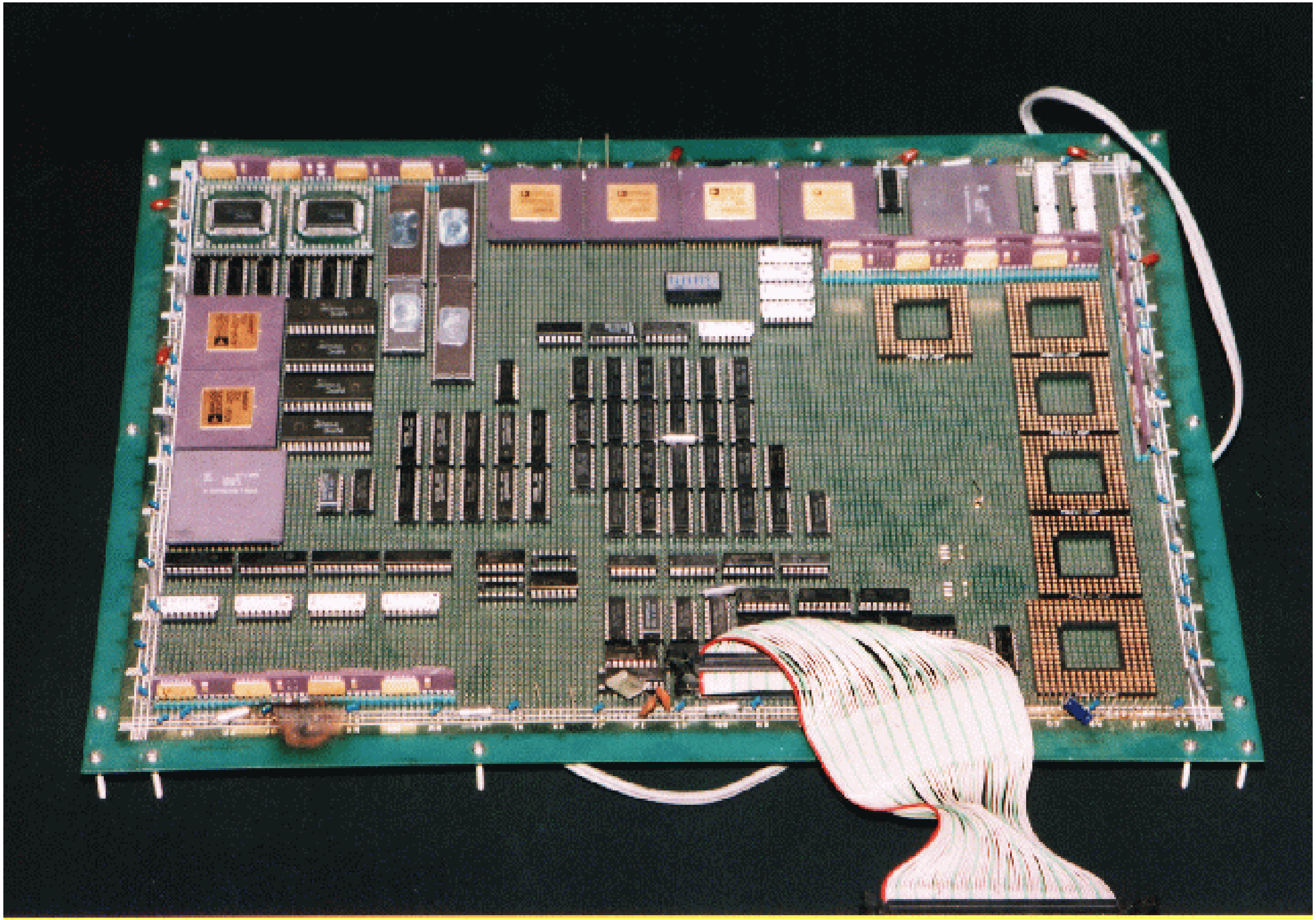


**Might be better known as the author of the comic series “Eiko-naki tensai tachi” (Geniuses without fame)
Now professor of EE at Chiba Univ.**

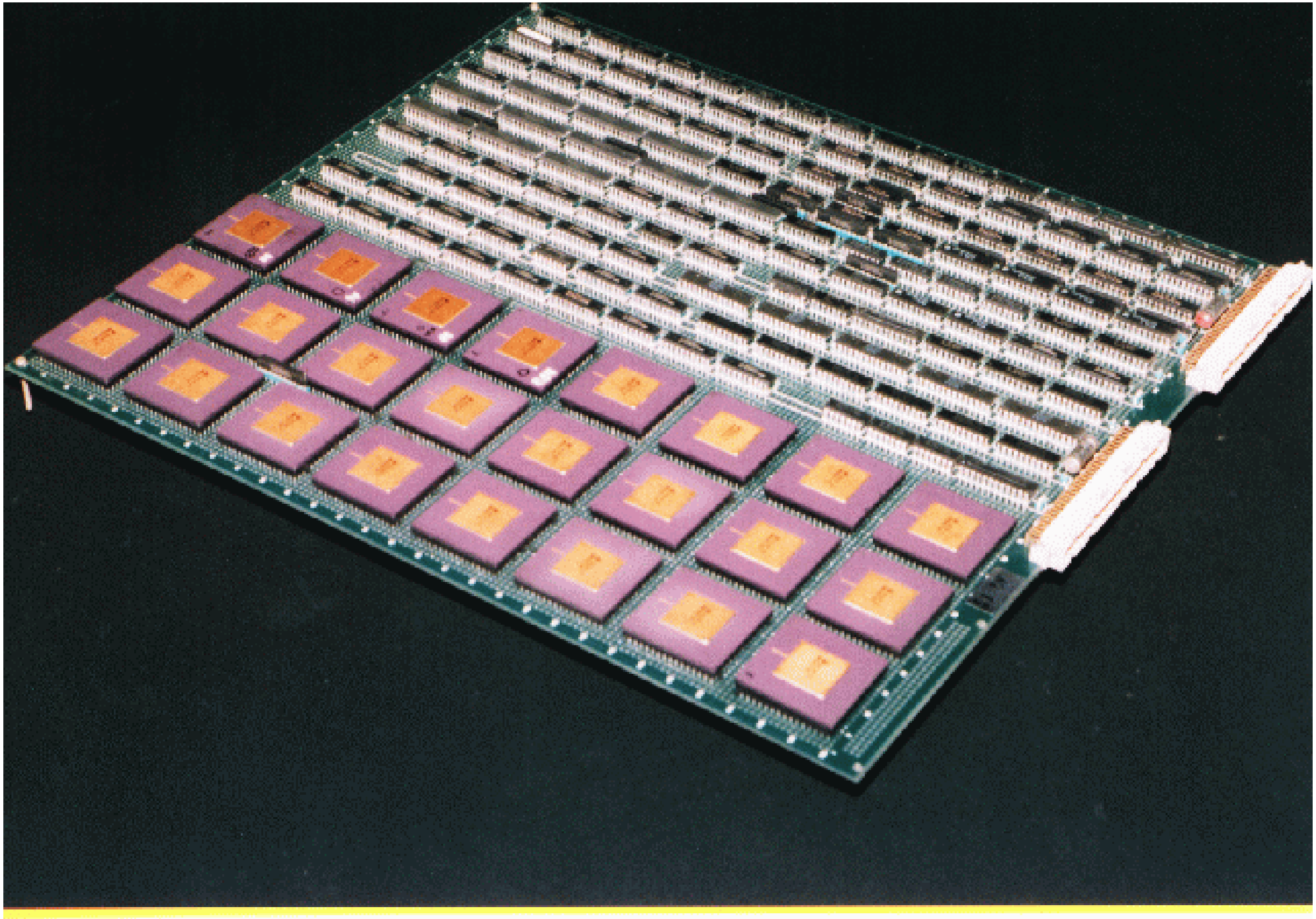
Lessons learned

- **Communication software is difficult**
- **“Recommended” or usual methods in textbook does not necessarily give good result**
- **Good result justifies whatever approach used**

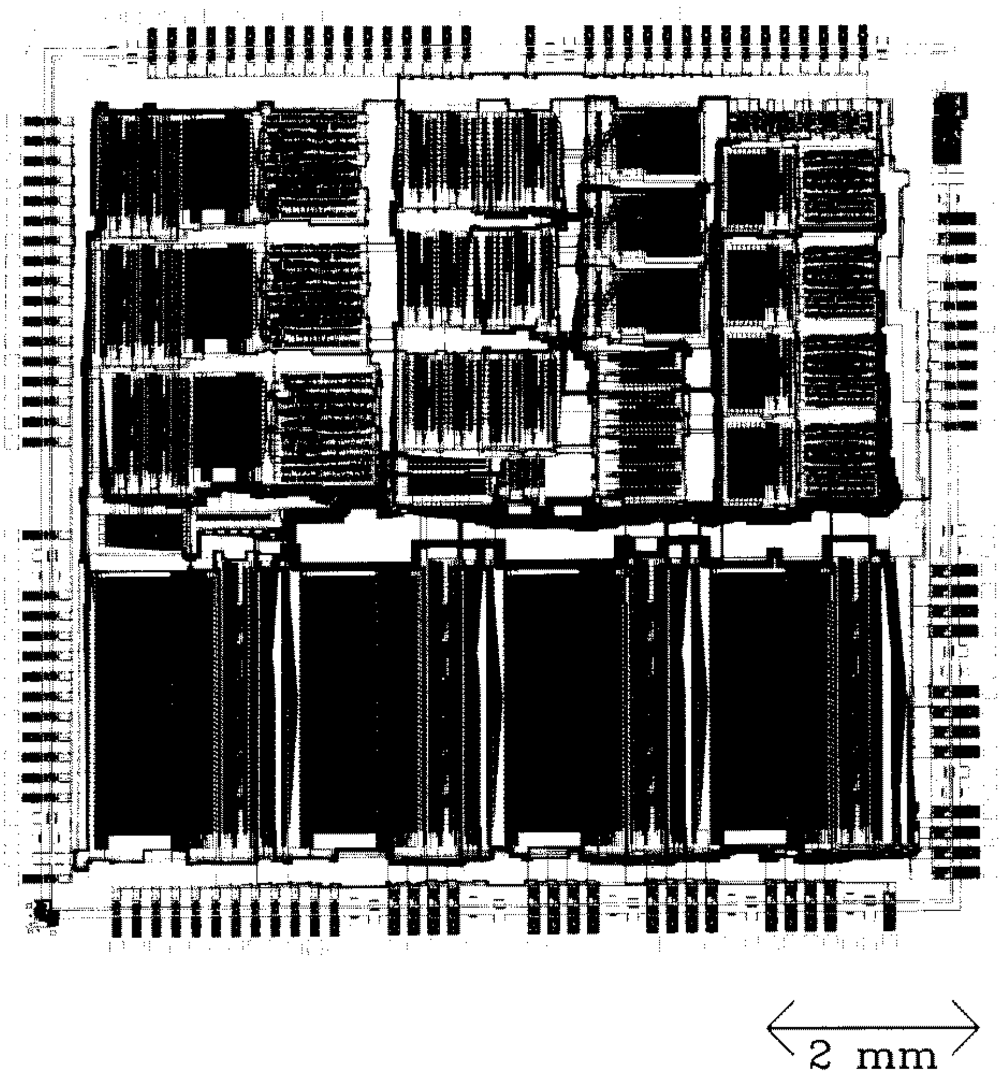
GRAPE-2



GRAPE-3



GRAPE-3 Custom LSI



GRAPE-3 chip design

- **Specification, behavior simulation: JM**
- **Detailed logic design: Fuji-Xerox**
- **SCS Genesil design tool**
- **National Semiconductor. 1μm**

How the chip-making affect your health?

We never had the budget for “respin”, or redesign of the chip

Division of the responsibility

- If the test pattern did not get through, that is manufacturer’s fault
- If other faults found, that’s my fault...

In theory, if we can prepare perfect test pattern, there will be no problem.

In practice...

GRAPE-4

GRAPE-6

- Design principle
- Processor chip
- Overview

Design principle

Goal of the project (when we got budget)

— achieve the world's best performance

Our real goal:

To build a machine which can do real sciences.

Boundary condition

- **Budget: 500MJYE (Earth Simulator 50BJYE, ASCI Q 200MUSD**

Target performance: 200Tflops (5 times that of ES)

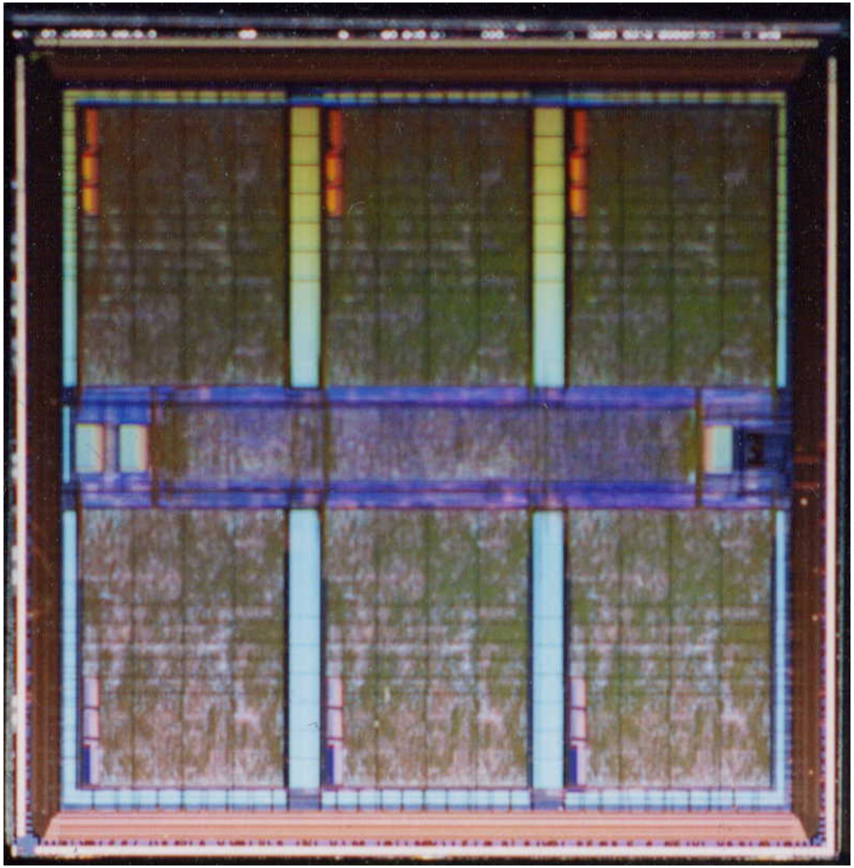
Performance prediction for GRAPE-6

Prediction: Extrapolation from GRAPE-4

	G4	G6 (pred)	G6 (real)
Design	1μm	0.25μm	0.25μm
Clock	32 MHz	125 MHz	90MHz
Pipelines	1/3	5-10	6
Performance	600Mflops	36-72 Gflops	31 Gflops
Initial Cost	25M	70M	More than 100M
Chip cost	8000K	10-20K	30K

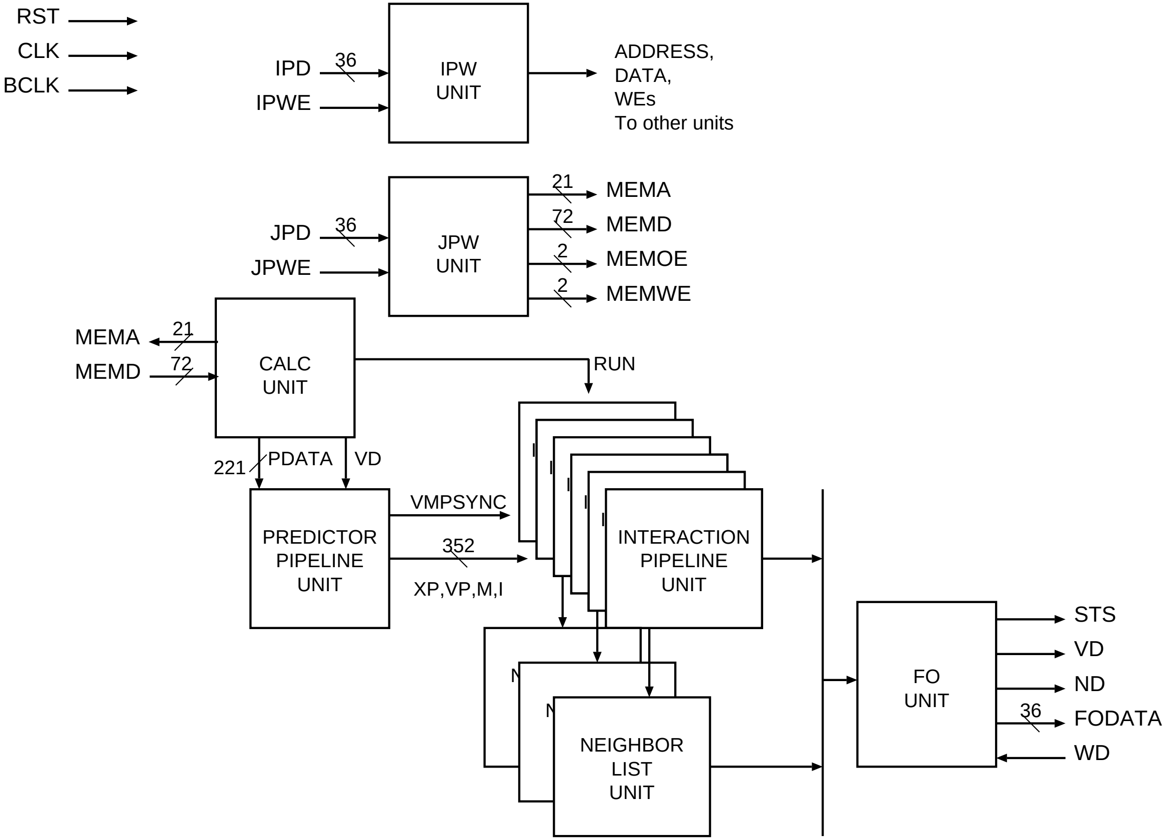
Accurate except for the cost estimate...

Pipeline chip



- **0.25 μm (Toshiba TC-240, 1.8M gates)**
- **90 MHz Clock**
- **6 pipelines**
- **31 Gflops**

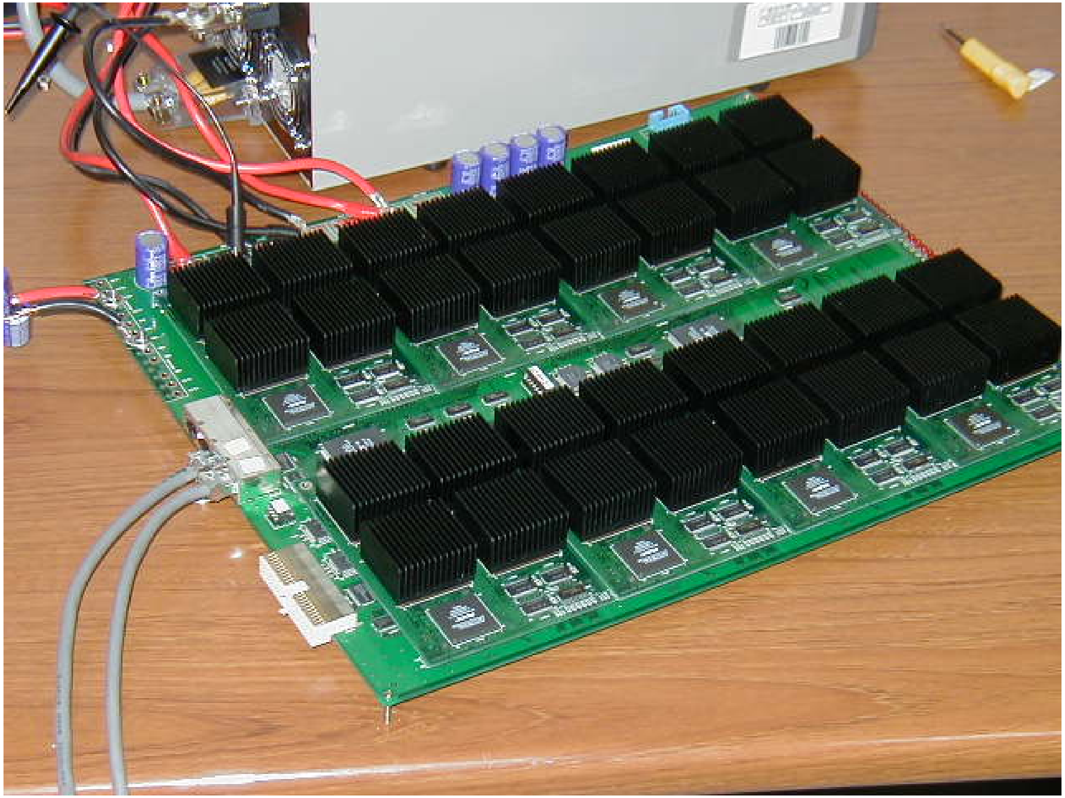
Details of LSI



Equivalent to single GRAPE-4 board

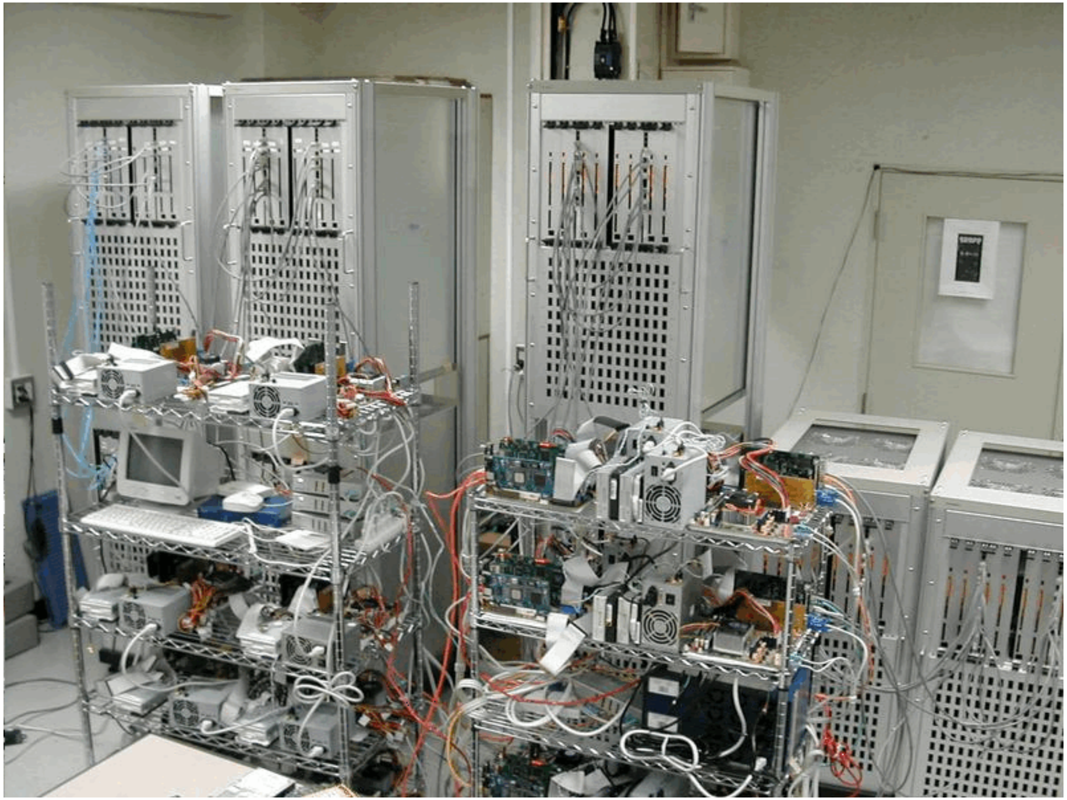
- Host IF
- Memory IF
- Pipelines
- Controls

GRAPE-6 processor board



- **32 chips/board**
- **LVDS interface(350MHz clock, 4 wires, about 1Gbps)**

The 64-Tflops GRAPE-6 system

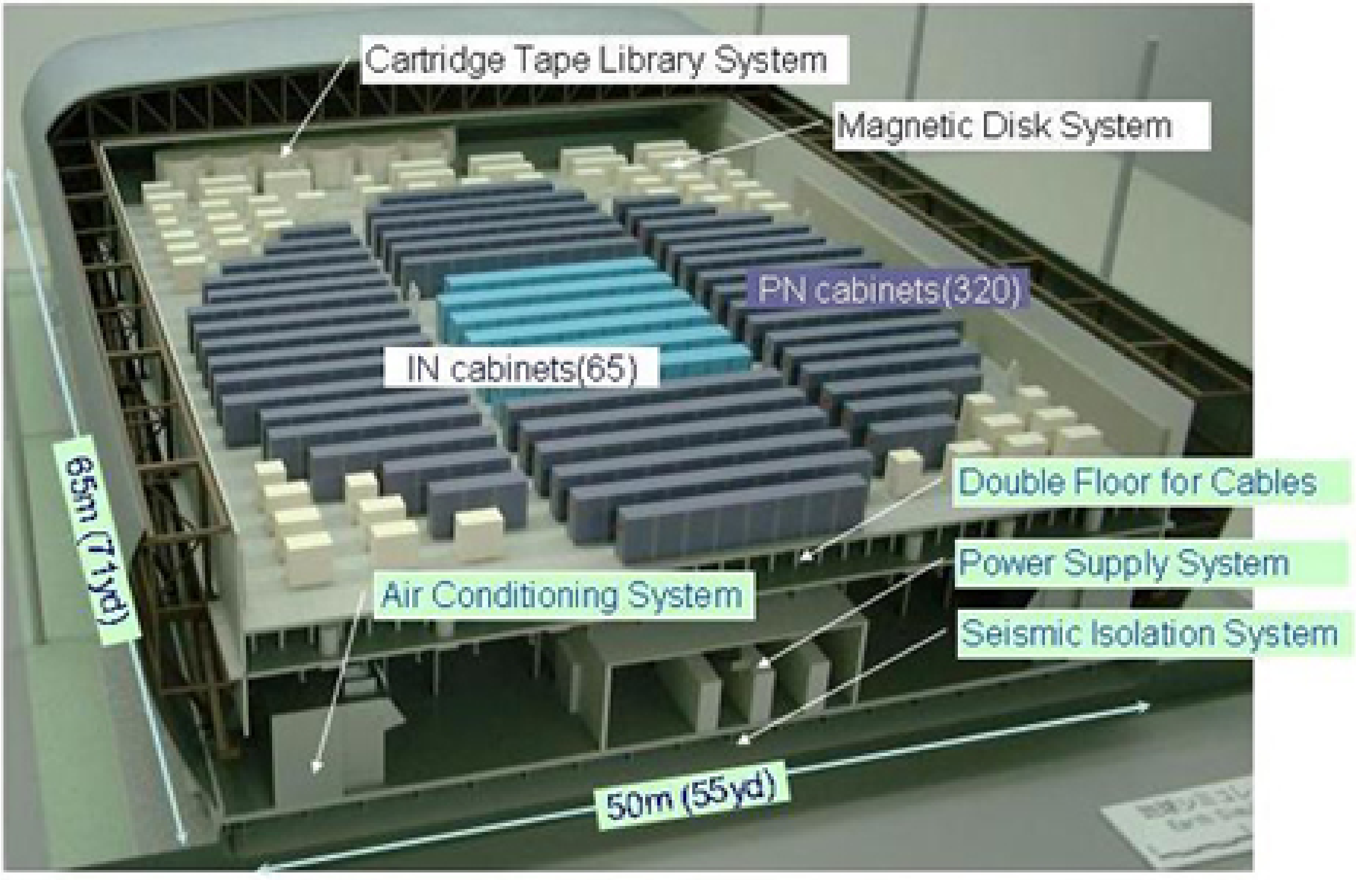


64-Tflops system.

4 blocks with 16 host computers.

In one room in Building 3, Asano-area of UT

The 40-Tflops Earth Simulator



Comparison with a recent Intel processor

	GRAPE-6	Intel Xeon 5365
Year	1999	2006
Design rule	250nm	65nm
Clock	90MHz	3GHz
Peak speed	32.4Gflops	48Gflops
Power	10W	120 W
Perf/W	3.24Gflops	0.4 Gflops

“Problem” with GRAPE approach

- **Chip development cost becomes too high.**

Year	**Machine**	**Chip initial cost**	**process**
1992	**GRAPE-4**	**200K$**	**1μm**
1997	**GRAPE-6**	**1M$**	**250nm**
2004	**GRAPE-DR**	**4M$**	**90nm**
2008?	**GDR2?**	**$\sim$ 10M$**	**65nm?**

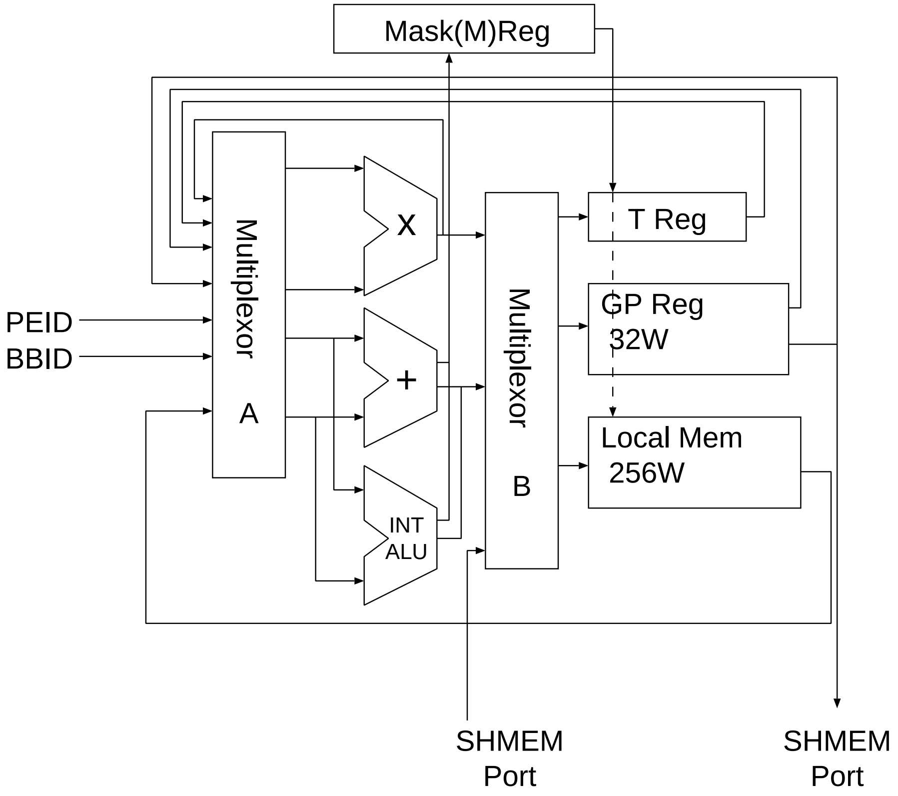
Initial cost should be 1/4 or less of the total budget.

How we can continue?

Next-Generation GRAPE — GRAPE-DR

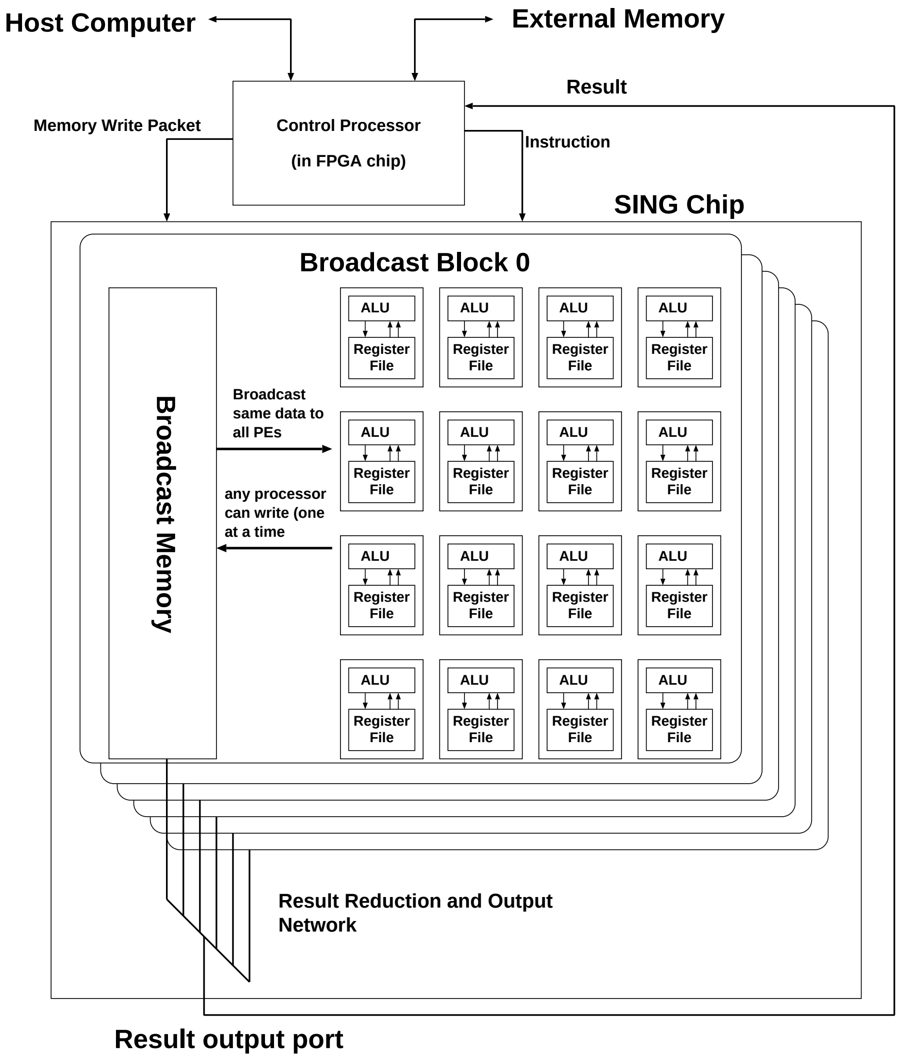
- **Planned peak speed: 2 Pflops**
- **New architecture — wider application range than previous GRAPEs**
- **primarily to get funded**
- **No force pipeline. SIMD programmable processor**
- **Planned completion year: FY 2008 (early 2009)**

Processor architecture



- **Float Mult**
- **Float add/sub**
- **Integer ALU**
- **32-word registers**
- **256-word memory**
- **communication port**

Chip structure



Collection of small processors.

512 processors on one chip
500MHz clock

Peak speed of one chip: 0.5 Tflops (20 times faster than GRAPE-6).

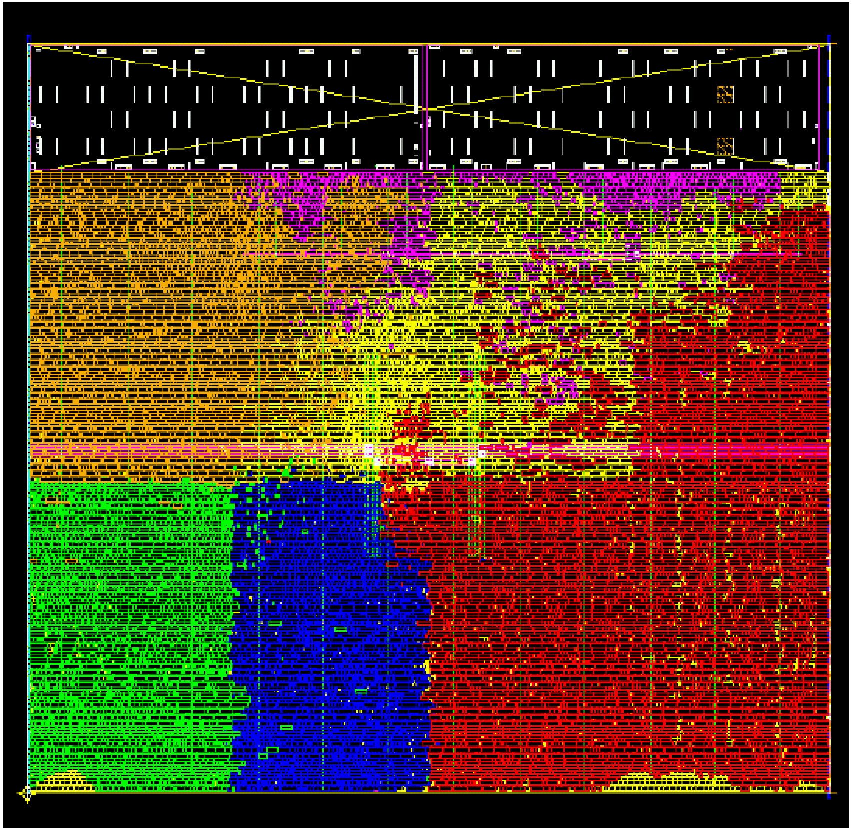
Why we changed the architecture?

- To get budget (*N*-body problem is too narrow...)
- To allow a wider range of applications
  - Molecular Dynamics
  - Boundary Element method
  - Dense matrix computation
  - SPH
- To allow a wider range of algorithms
  - FMM
  - Ahmad-Cohen
  - ...

Development status

Sample chip delivered May 2006

PE Layout



0.7mm by 0.7mm

Black: Local Memory

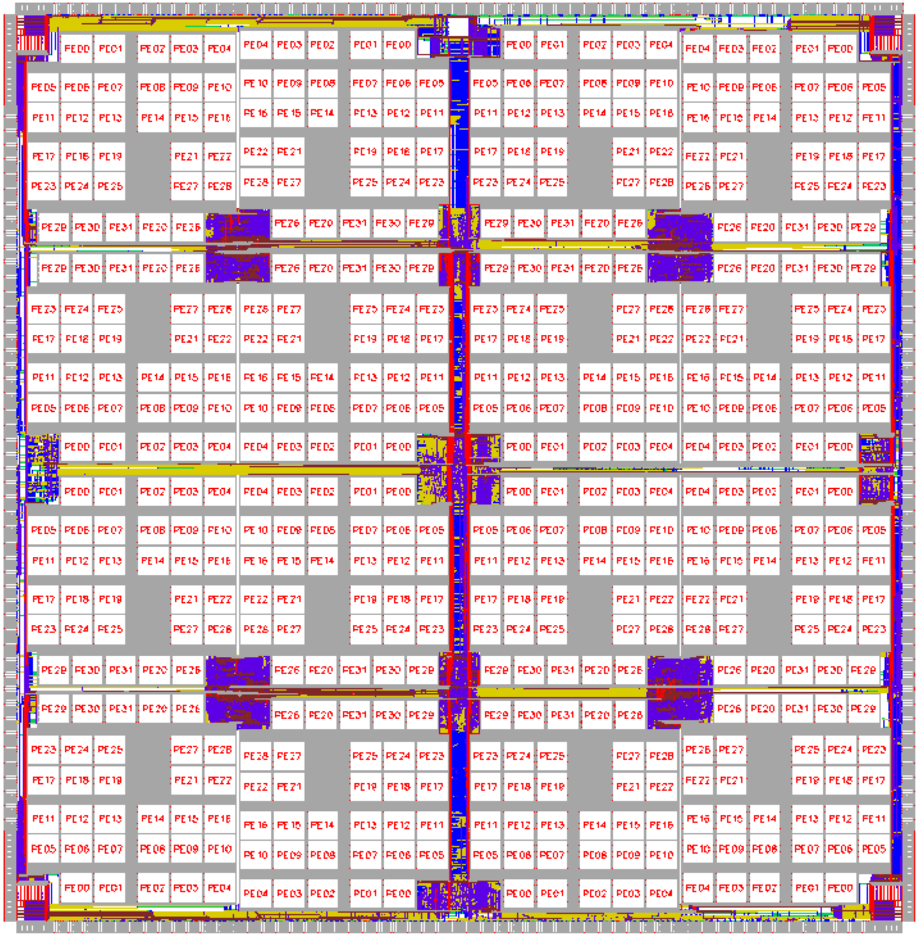
Red: Reg. File

Orange: FMUL

Green: FADD

Blue: IALU

Chip layout



- **32PEs in 16 groups**
- **18mm by 18mm**

Prototype board

2nd prototype. (Designed by Toshi Fukushige)
Single-chip board
PCI-Express x8 interface
On-board DRAM
Designed to run real applications
(Mass-production version will have 4 chips)

GDR-2?

- **We are trying hard to get some money from Japan's "Next-Generation Supercomputer Project"**
- **With 65nm, it is not difficult to achieve**
  - **768 DP Gflops/chip**
  - **1.5 SP Tflops/chip**
  - **On-chip memory (16-32MB)**

Japan's Next-generation Supercomputer Project

- **FY 2006-2012**
- **Total budget: 110 BJYE (about 80 times that of GRAPE-DR)**
- **Peak speed: 10Pflops (about 10 times that of GRAPE-DR)**
- **Vector (like ES) + Scalar (???) hybrid**

Summary

- GRAPEs seems to be fairly successful
- However, we cannot continue...
- With GRAPE-DR, we moved to new architecture
- We'll see if this was the right move or not.

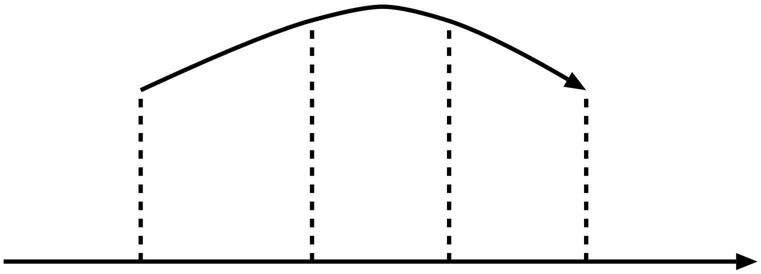
Integration scheme

Integration order: "4th the best" (JM 1990)

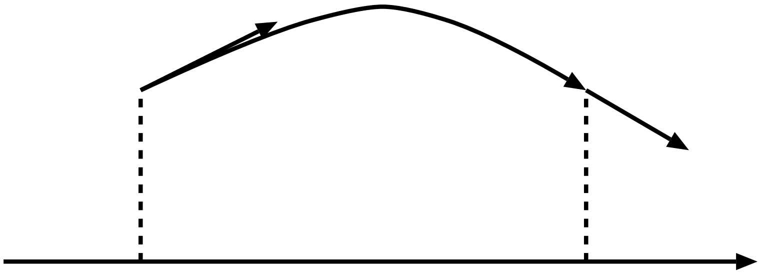
6-8th seems better: (Nitadori and JM 2007)

- **Aarseth scheme (Aarseth 1963): Adams scheme, PEC mode, 4th**
- **Hermite scheme (JM 1990): Hermite interpolation with direct calculation of the first time derivative of the force**

Aarseth scheme and Hermite scheme



Lagrange
(Newton)



Hermite

Left: Aarseth scheme with Newton interpolation

Right: Hermite scheme

Hermite scheme is much simpler to implement and faster

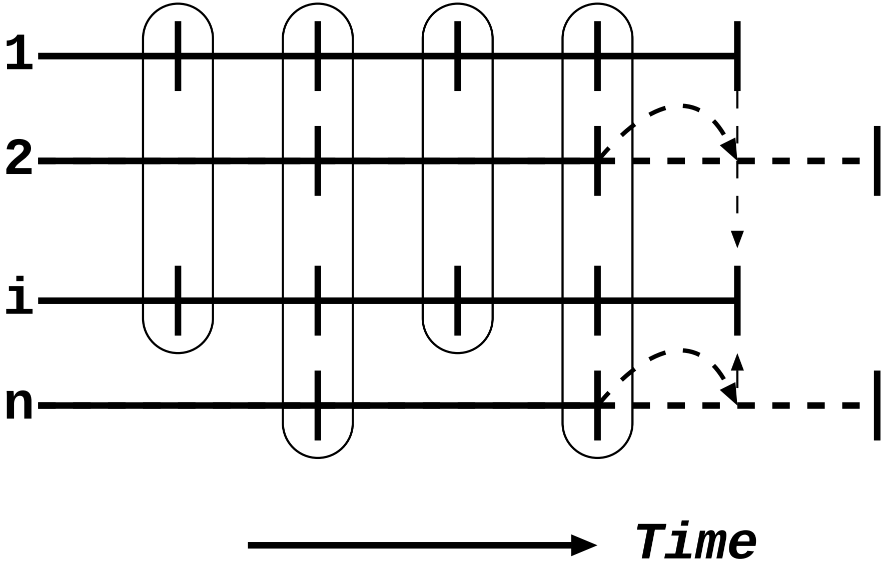
High-order Hermite schemes

Nitadori and JM 2007

- Direct calculation of second derivative: 6th
- Direct calculation of third derivative: 8th
- Predictor requires the values at previous steps (for 4th order scheme previous value was not needed)

Block individual timesteps

(McMillan 1986) improvement for parallel computers



- Limit timesteps to 2^{-k}
- stars with the same $t_i + \Delta t_i$ (even with different Δt_i) integrated in parallel

$O(N_c^{2/3})$ stars (N_c: number of stars in the core)

Initial plan

- ROM table for arithmetic, 8-bit data
- GPIB (IEEE-488) communication

Calculation/communication: Designed to use with 10,000 particles or around

5 MHz clock = can use around 1 ms for one particle communication

$<$ 100 bytes data per particle $\rightarrow$ communication speed 100KB/s

RS-232C too slow

SCSI too difficult to design

Design change

It turned out that, even if the accuracy of the pairwise force is low, if we do

- **first subtraction of positions**
- **final accumulation of the force**

in high accuracy, we can achieve the accuracy better than treecode.

First sub: 16bit, final acc:48bit, both fixed point

Personal feeling...

It seems rather difficult to build something like GRAPE
Understandings of

- Target problem
- Algorithm, accuracy
- Computer architecture
- Digital logic design
- Physical design (packaging, cooling....)
- OS, device driver etc

need to be integrated, ideally in a single person.

(Digital Orrery: G. Sussman had all of this)

Not necessary the world best understanding, though.
Something reasonable is okay.

Compared to general purpose computer

With general-purpose computer you don't have to worry about

- **Target problem**
- **Algorithm, accuracy**

really?

- **Other things:**

You need the best.

Are special-purpose computers difficult?

Rather few successful examples.

Particle systems: two approaches

Pipelined processors

- DMDP (Delft)
- FASTRUN
- GRAPE
- MD-Engine
- MDM

Programmable Parallel machines

- Digital Orrery
- Transputer-based projects...
- HaMM
- Many others

Why failures?

Two reasons:

1. **Machine could not be used**
2. **Machine too slow when completed**

Second one is much more common.

Problem with development time

Almost everybody (including myself) is too optimistic.

Essential problem:

In the case of special-purpose computer, a project which loses meaning with 1-2 years of delay should not be started.

Roughly speaking, when you start, if the price performance is better by

- 1000 — okay
- 100 — getting difficult
- 10 — should not start

If we assume five-year development time and five-year lifetime of the hardware.

A few more words on software

- **The right way to separate the task between host CPU and (GRAPE, GRAPE-DR, GPU, FPGA) is the same**
- **The right way to make efficient use of large number of processors on (GRAPE, GRAPE-DR, GPU, FPGA, CPU) is the same**

We should develop a common software platform for different hardwares

Preliminary data for first commercial version

- Prototype board working
- 1 Chip on a board (0.5Tflops peak)
- PCI-Express x4 interface
- 80W ...
- $\sim$ 5K USD ...

Dynamical time

For a stellar system with mass M, typical radius R, we have the Virial Theorem

$$E = -K = W/2$$

E: total energy ($K + W$), K: total kinetic energy, W: total potential energy,

$$\begin{aligned} W &= \sum_{i<j} G\frac{m_i m_j}{|x_i - x_j|} \\ K &= \sum_i \frac{1}{2} m_i v_i^2 \end{aligned}$$

Dynamical time (2)

For R, we have

$$W \sim -\frac{GM^2}{R}$$

and for K

$$K = -\frac{Mv^2}{2}$$

and we have $v \sim \sqrt{GM/R}$, and

$$T = R/v = \sqrt{\frac{R^3}{GM}}$$

Nonexistence of the thermal equilibrium

Thermal equilibrium, if exists, must be described by the Maxwell-Boltzmann statistics.

This is however impossible for a stellar system.

Reason:

Energies of all stars in the system cannot exceed the potential energy at the infinity (otherwise they go to infinity).

Therefore, there must be an upper cutoff in the energy distribution function.

Final state of stellar systems

Essentially the same as $N = 3$.

If high-energy stars are generated through gravitational scattering, they escape from the system.n

In other words, from the equilibrium statistical mechanics we can conclude:

Every gravitational many-body system will evaporate, if we wait long enough

This is certainly true, but not too useful for the understanding of existing stellar systems.

We do need non-equilibrium statistical mechanics.

Principle of the individual timestep

Each star has its own time $\boldsymbol{t_i}$ and timestep $\boldsymbol{\Delta t_i}$

1. **Select the star with minimum $\boldsymbol{t_i + \Delta t_i}$**
2. **Integrate its orbit to its new time $\boldsymbol{t_i + \Delta t_i}$**
3. **determine its new timestep**
4. **go back to step 1.**

We need high-accuracy position prediction for other stars at time $\boldsymbol{t_i + \Delta t_i}$.

Predictor-corrector type schemes are used.

Calculation cost and accuracy

Simple estimate:

$$\boldsymbol{error \propto \theta^{(p+1)}}$$
$$\boldsymbol{cost \propto \theta^{-3} p^2 N \log N}$$

p: **expansion order**

$\log N$: **tree level**

θ^3: **number of cells in one level which interact with one particle**

In reality...

Actual behavior: rather complex

- **Accuracy is better than the estimate in the previous slide**
- **Calculation cost shows weaker dependence on θ**

Calculation cost for thermal evolution

- **per step:** N^2
- **number of orbits:** $N/\log N$
- **steps per orbit:** $> N^{1/3}$
- **In total:**

$$\frac{N^{10/3}}{\log N}$$

- $N \sim 2 \times 10^5$ **is the current limit with fastest computers available**

Numerical integration over thermal timescale

- Very costly
- Do we need to do such expensive calculations?
- Can't we rely on some statistical approach, if the system is statistical anyway?

I do not have a short answer...

Summary of a long answer

- **Thermal equilibrium does not exist**
- **Small-N effects alway become important**

As a result:

Reliable statistical methods are very difficult to construct

Numerical methods

Let me discuss the techniques for numerical integration.

Very naively, it is important to do calculations

- **with large N**
- **with high accuracy**
- **for long time**

since that helps to develop the better understanding.

How we can do better calculations?

Basic idea: If we can do the same calculation faster, that means we could use larger N or achieve higher accuracy, if we use the same amount of the computer time

Impact on the calculation cost

Simple variable timestep would cost too much

Reason: Power-law distribution of timesteps

Calculation cost increases as some power of N
Structure formation: $O(N^{1.3})$ or around
Two-body scattering: $O(N^{1/3})$

Solution:

- **Assign different timesteps to different stars (individual timestep)**
- **Two-body collision, binaries: Coordinate transformation**

Memory bandwidth requirement

Reduction of communication

Host — GRAPE: N stars, N^2 calculation

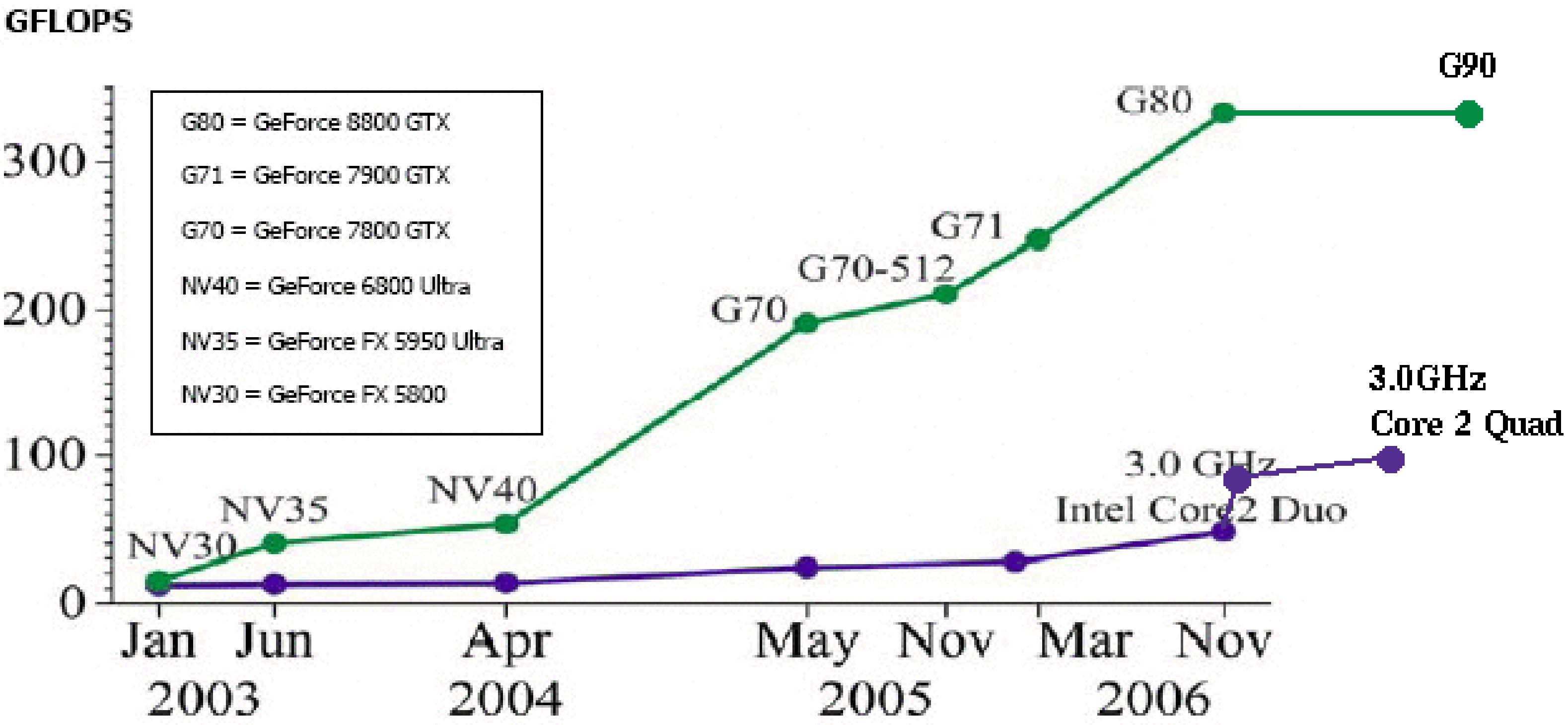
Board/chip level:

- **Multiple pipelines calculate the force from same particle to different particles**
- **Virtual multiple pipeline: One pipeline calculates the forces on several particles**

Comparison with FPGA

- **much better silicon usage (ALUs in custom circuit, no programmable switching network)**
- **(possibly) higher clock speed (no programmable switching network on chip)**
- **easier to program (no VHDL necessary; assembly language and compiler instead)**

GPGPUs —Today



Hmm...

How do you use it?

- GRAPE: The necessary software is now ready. Essentially the same as GRAPE-6.
- Matrix etc ... RIKEN/NAOJ will do something
- New applications:
  - Primitive Compiler available
  - For high performance, you need to write the kernel code in assembly language (for now)

Primitive compiler

(Nakasato 2006)

```
/VARI  xi, yi, zi, e2;
/VARJ  xj, yj, zj, mj;
/VARF  fx, fy, fz;
dx = xi - xj;
dy = yi - yj;
dz = zi - zj;
r2 = dx*dx + dy*dy + dz*dz + e2;
r3i= powm32(r2);
ff = mj*r3i;
fx += ff*dx;
fy += ff*dy;
fz += ff*dz;
```

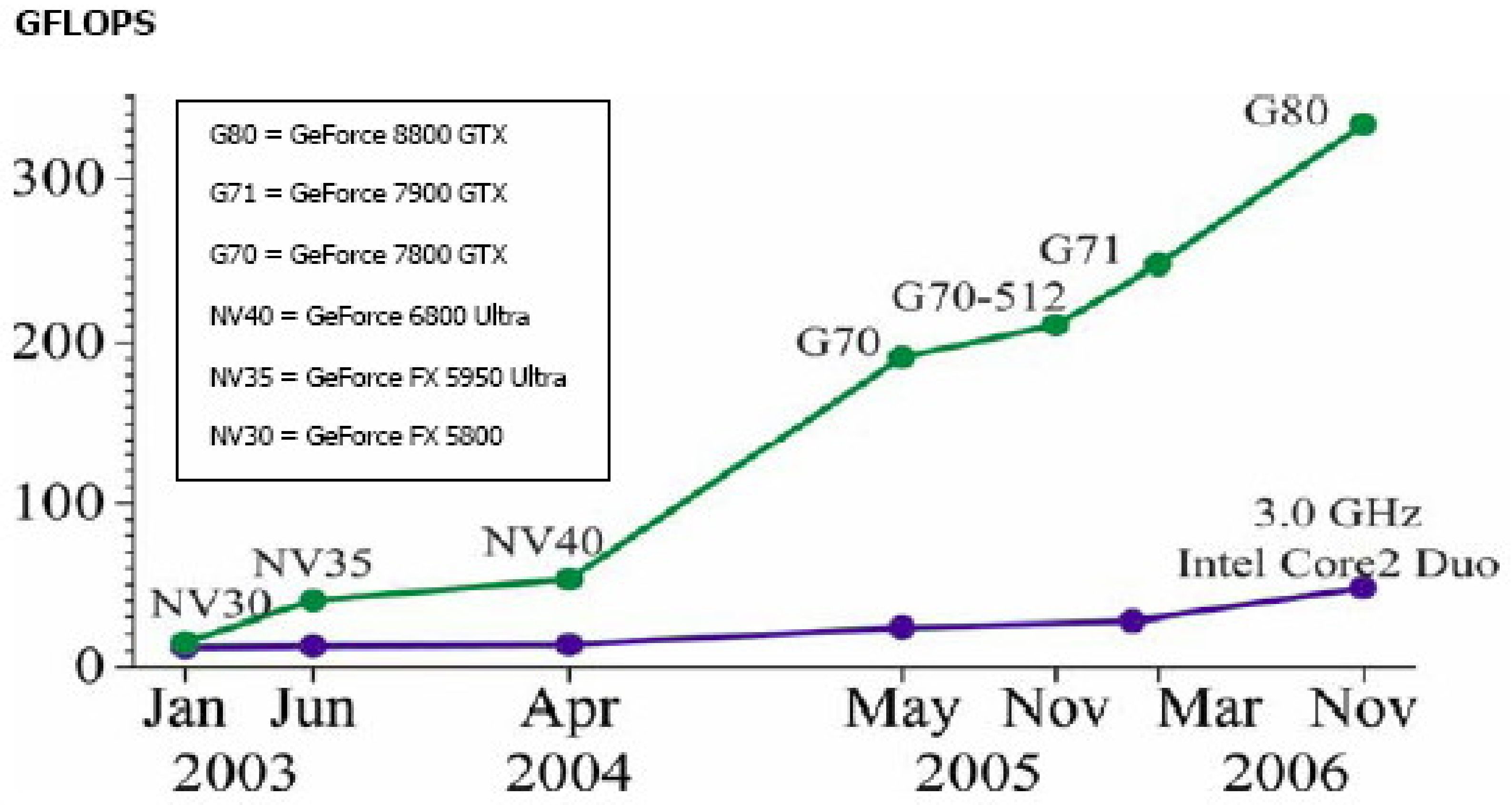
- **Assembly code**
- **Interface/driver functions**
- **SIMD parallel data distribution**
- **Data reduction**

are generated from this ”high-level description”.
(Can be ported to GPUs)

Interface functions

```
struct SING_hlt_struct0{
  double xi;
  double yi;
  double zi;
  double e2;
};
int SING_send_i_particle(struct SING_hlt_struct0 *ip,
                         int n);
int SING_send_elt_data0(struct SING_elt_struct0 *ip,
                        int index_in_EM);
int SING_get_result(struct SING_result_struct *rp);
int SING_grape_run(int n);
```

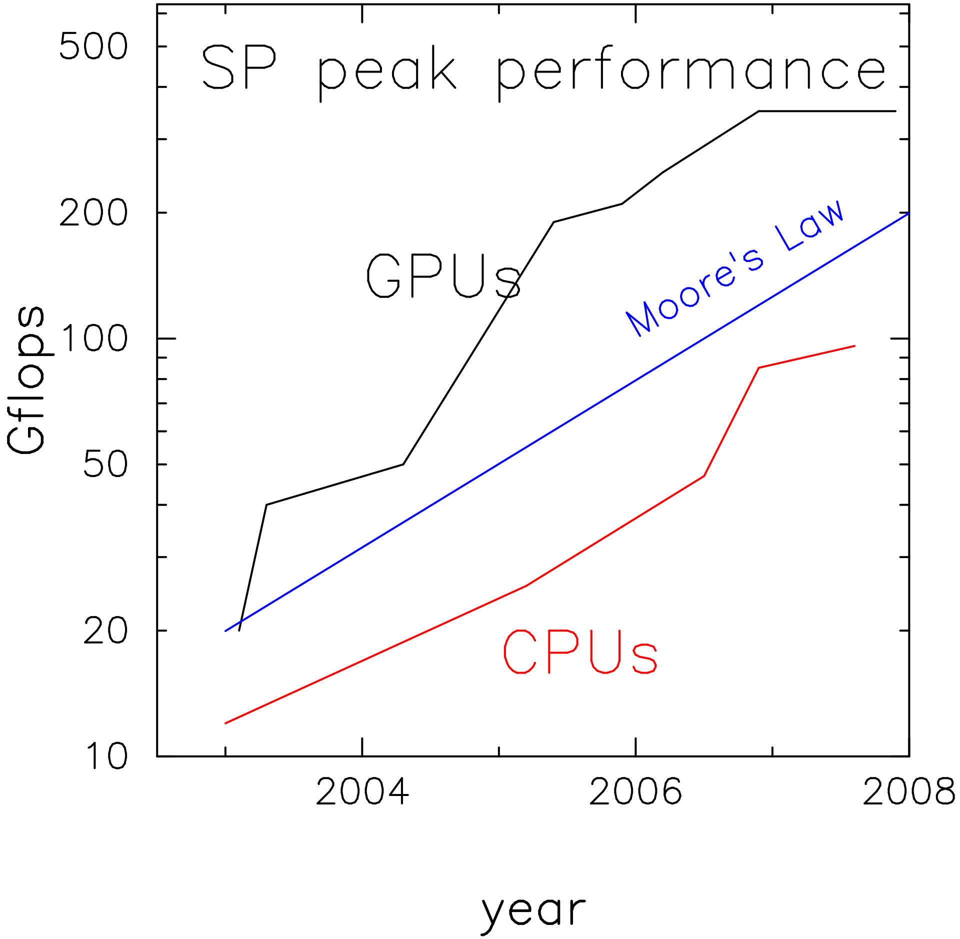
GPGPUs —What manufacturers show:



“GPUs beat Moore’s Law!”

(AstroGPU, Nov 9-10, 2007, IAS, Princeton)

GPGPUs —Same data in log plot



- **Faster-than-Moore period ended in 2005**
- **Microprocessors are catching up**
- **DP performance?**
- **Design limit with memory bandwidth**

Communication overhead

NEWS was very slow

Reason: GPIB communication is through UNIX OS system call, which incurred more than 1ms overhead.

Our initial approach: Use NEC PC for buffering the data

Final solution: hack the operating system and let the application program directly manipulate the GP-IP controller LSI.