GPU Accelerated Monte Carlo Simulations of Dense Stellar Systems

Bharath Pattabiraman

Northwestern University,

Evanston, USA

Team: Stefan Umbreit, Wei-keng Liao, Frederic Rasio, Vassiliki Kalogera, Gokhan Memik, and Alok Choudhary

Agenda

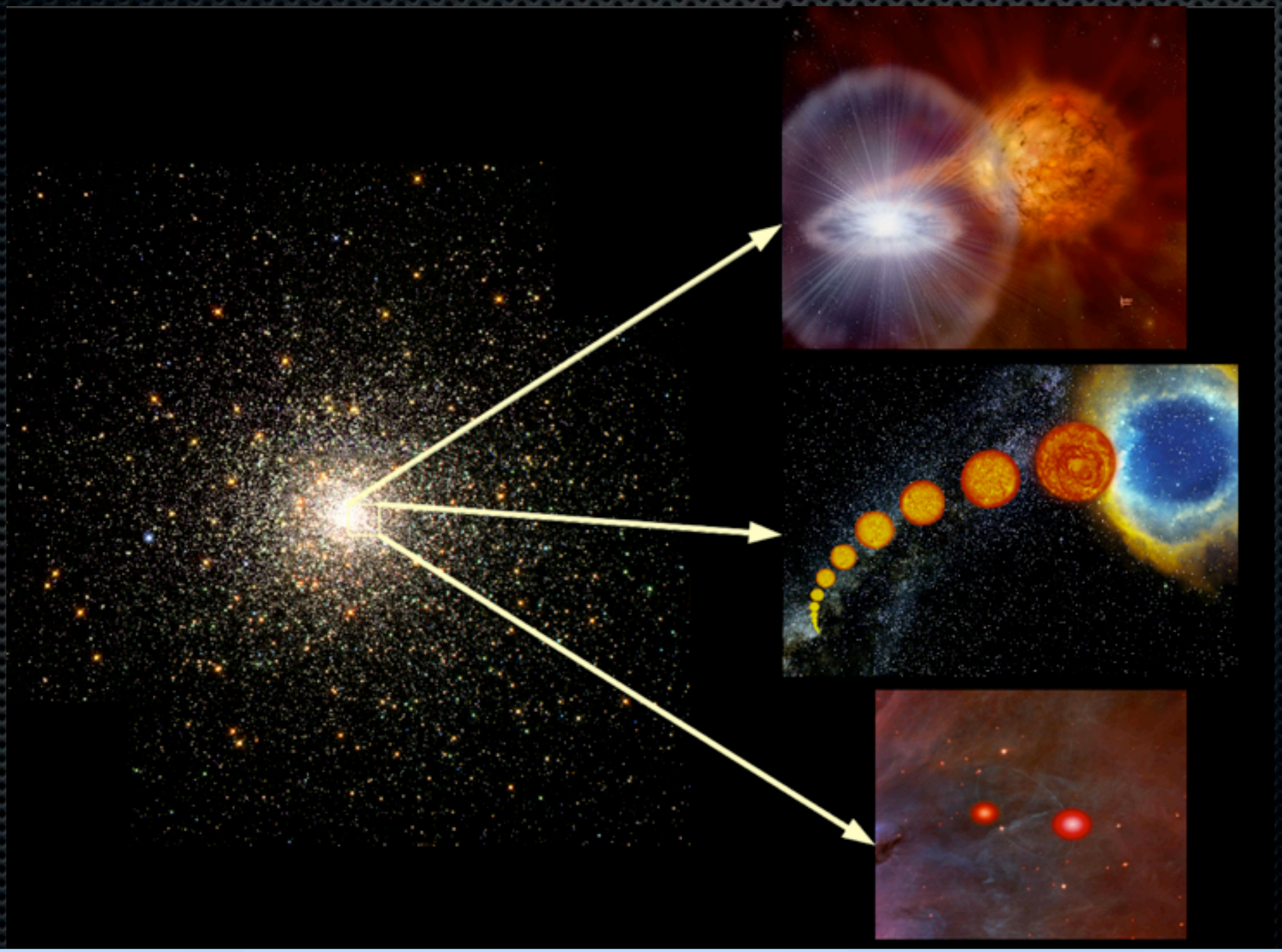
- **Dense Stellar Systems**
- The Monte Carlo Method
- Performance Bottlenecks
- GPU Implementation
- Results

Dense Stellar Systems

- **Globular Cluster**
  ~ **10^5 to 10^7** stars
- **Galactic Nuclei**
  ~ **10^9** stars

(NASA/ESA 1999)

Multi-physics & Multi-scale



(Figures: ESO/S. Steinhöfel, NASA, The Hubble Heritage Team, STScI, AURA)

Dense Stellar Systems Modeling

- **Direct N-body**
  - Free of any physical assumptions (for the N modeled)
  - Direct computation of inter-particle forces
  - Poor Scaling **O(N^2)** => Extreme Computing time
  - Maximum N ~ few 10^5
- **Monte Carlo**
  - Approximate, based on two-body relaxation
  - Better scaling: **O(NlogN)** => Much Faster
  - Maximum N > 10^6

Agenda

- Dense Stellar Systems
- **The Monte Carlo Method**
- Performance Bottlenecks
- GPU Implementation
- Results

The Monte Carlo Method

- **Assumptions**
  - Stars represented by mass shells
    - Position **r**, Mass **m**
  - Velocity specified by $\mathbf{v}_r$, $\mathbf{v}_t$ only
    - Angular coordinate of transverse velocity random
  - System is exactly spherically symmetrical

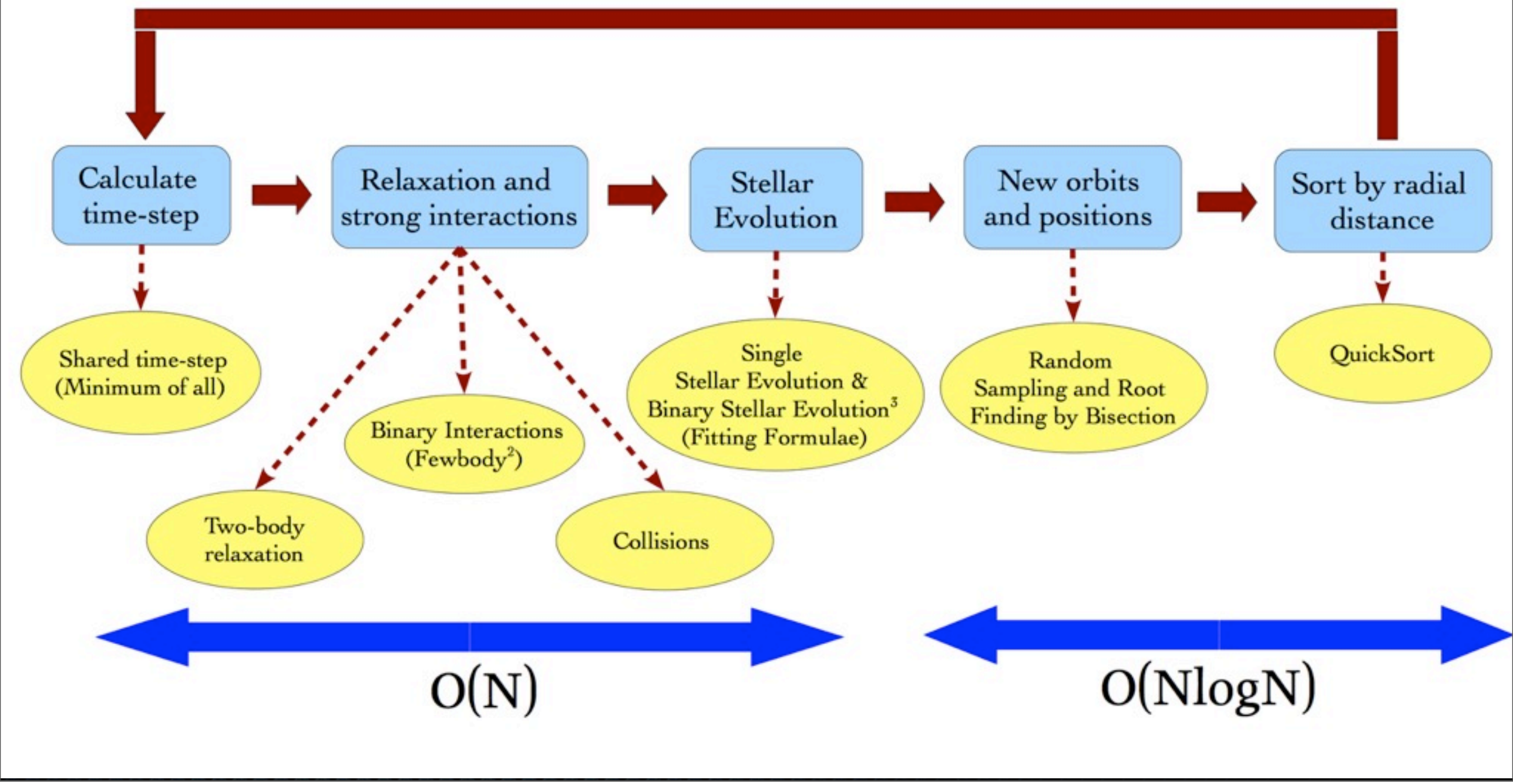
(Henon., 1971)

“Cluster Monte Carlo (CMC)” Code

- **Main Developer**: John Fregeau
- **Originally written by**: Kriten Joshi in 2000
- **Contributers**:

  Cody Nave
  Atakan Guerkan
  Stefan Umbreit
  Sourav Chatterjee
  Paul Kiel
  Bharath Pattabiraman
  Meagan Morscher

CMC Overview



Agenda

- Dense Stellar Systems
- The Monte Carlo Method
- **Performance Bottlenecks**
- GPU Implementation
- Results

Performance Profiling

Kernel	% Time
Time-step calculation	9%
Relax. and strong interaction	12%
Stellar Evolution	13%
New orbits and positions	54%
Sorting	7%
Other	5%

- CMC simulation of about 10^6 stars (virial radius ~ 3 - 4 pc) up to 10^{10} years takes of the order of a week
- For extreme cases, **more than a month**!

Potential Calculation

- All stars treated as mass shells
- Gravitational Potential:

$$U(r) = KG\left(-\frac{1}{r}\sum_{i+1}^{k} m_i - \sum_{i=k+1}^{n}\frac{m_i}{r_i}\right).$$

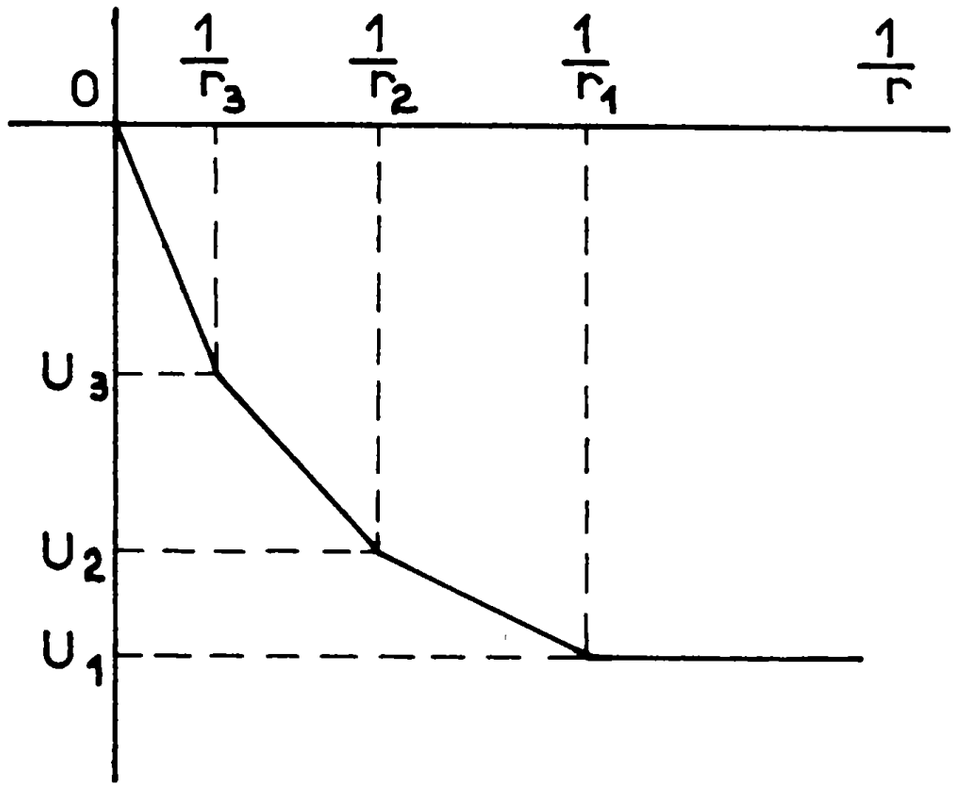


Fig. 1. Gravitational potential U as a function of $1/r$, for a system of three shells.

(Henon., 1971)

Potential Calculation

- It is sufficient to store the values $U_k = U(r_k)$ which can be calculated recursively:

$$
\begin{aligned}
&U_{n+1} = 0, \\
&M_n = M = K \sum_{i=1}^{n} m_i, \\
&\left.\begin{aligned}
U_k &= U_{k+1} - GM_k\left(\frac{1}{r_k} - \frac{1}{r_{k+1}}\right), \\
M_{k-1} &= M_k - Km_k .
\end{aligned}\right\} \quad (k = n, \ldots, 1)
\end{aligned}
$$

- Potential between shells interpolated

$$U(r) = U_k + \frac{1/r_k - 1/r}{1/r_k - 1/r_{k+1}}(U_{k+1} - U_k), \quad \text{for} \quad r_k \leqslant r \leqslant r_{k+1}$$

(Henon., 1971)

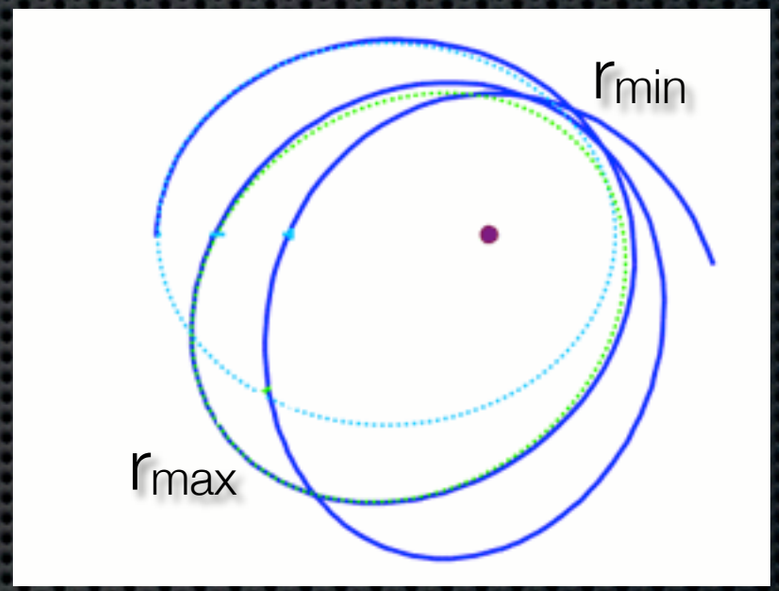
New Orbits Computation

- Star describes rosette orbit going from r_{min} to r_{max}
- Apsides solution of

$$Q(r) = 2E - 2U(r) - A^2/r^2 = 0.$$

- Potential at arbitrary point can be interpolated given the interval

$$U(r) = U_k + \frac{1/r_k - 1/r}{1/r_k - 1/r_{k+1}} (U_{k+1} - U_k), \quad \text{for} \quad r_k \leqslant r \leqslant r_{k+1}$$



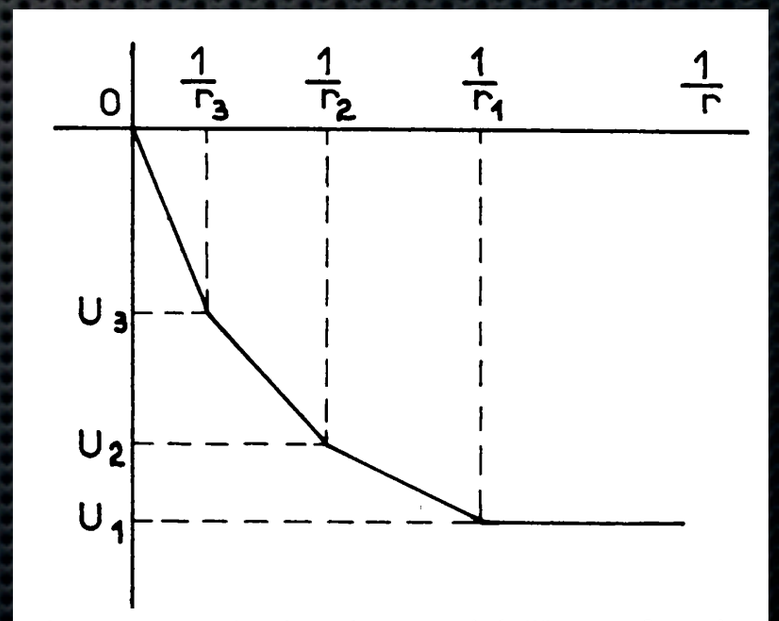


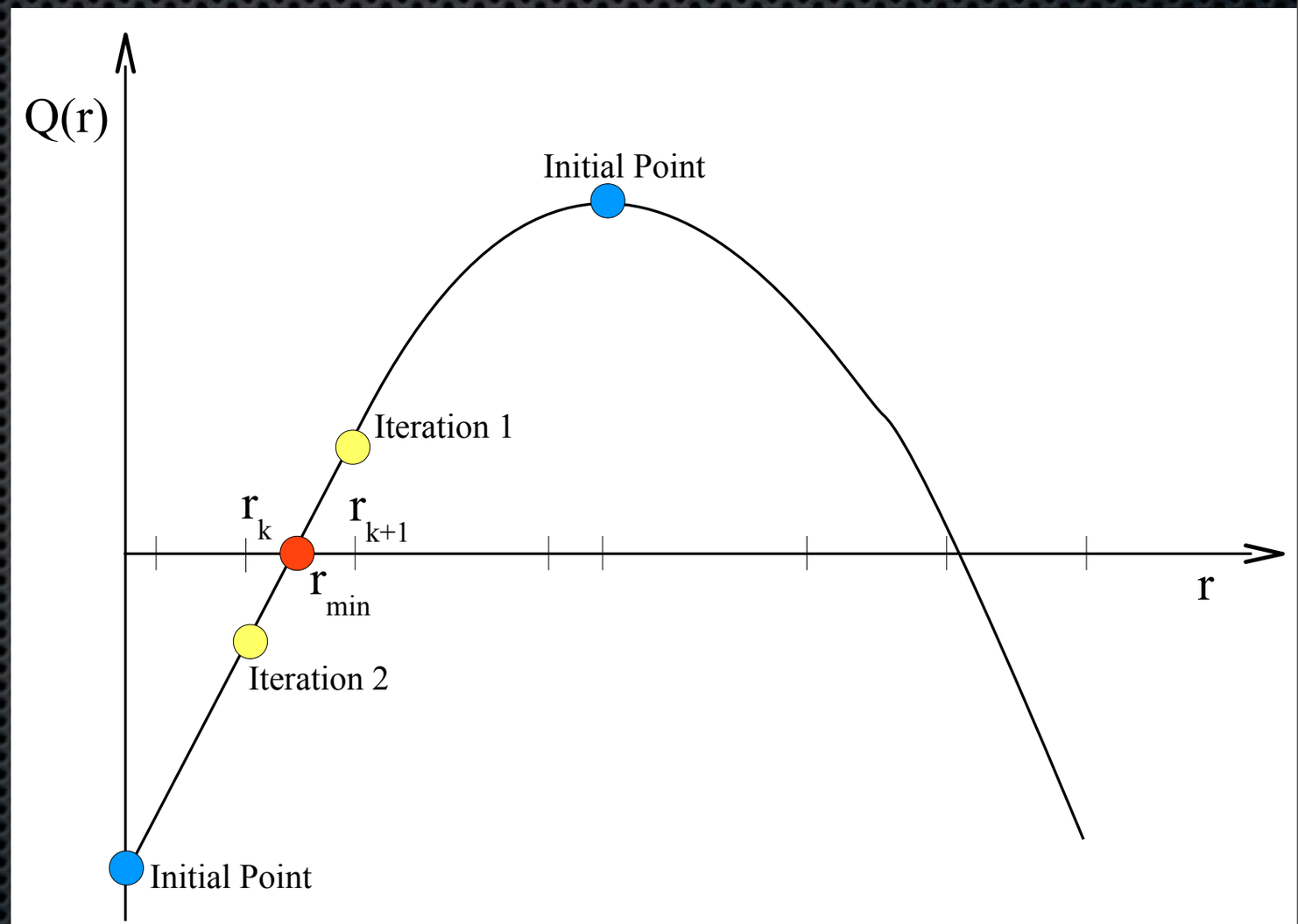
Fig. 1. Gravitational potential U as a function of $1/r$, for a system of three shells.

New Orbits Computation

- First, search interval containing zero (Root bisection)

$$Q(r_k) < 0 < Q(r_{k+1}).$$

- Solve for quadratic equation $Q(r) = 0$ to get r_{min} and r_{max}
- O(NlogN)



New Positions Sampling

- Choose a new position based on the amount of time a star spends at different points along the orbit.
- The probability of finding a star in a position **r** along the orbit is inversely proportional to $\mathbf{v_r}$

$$f(r) = 1/|v_r|$$

Von Neumann Rejection Sampling

(Hammersley and Handscomb., 1964)

Rejection Sampling

- Pick $F > f(r)$ for $\forall\ r_{min} < r < r_{max}$
- Pick 2 random numbers X and X'

$$r_0 = r_{\min} + (r_{\max} - r_{\min})\, X$$

$$f_0 = FX'$$

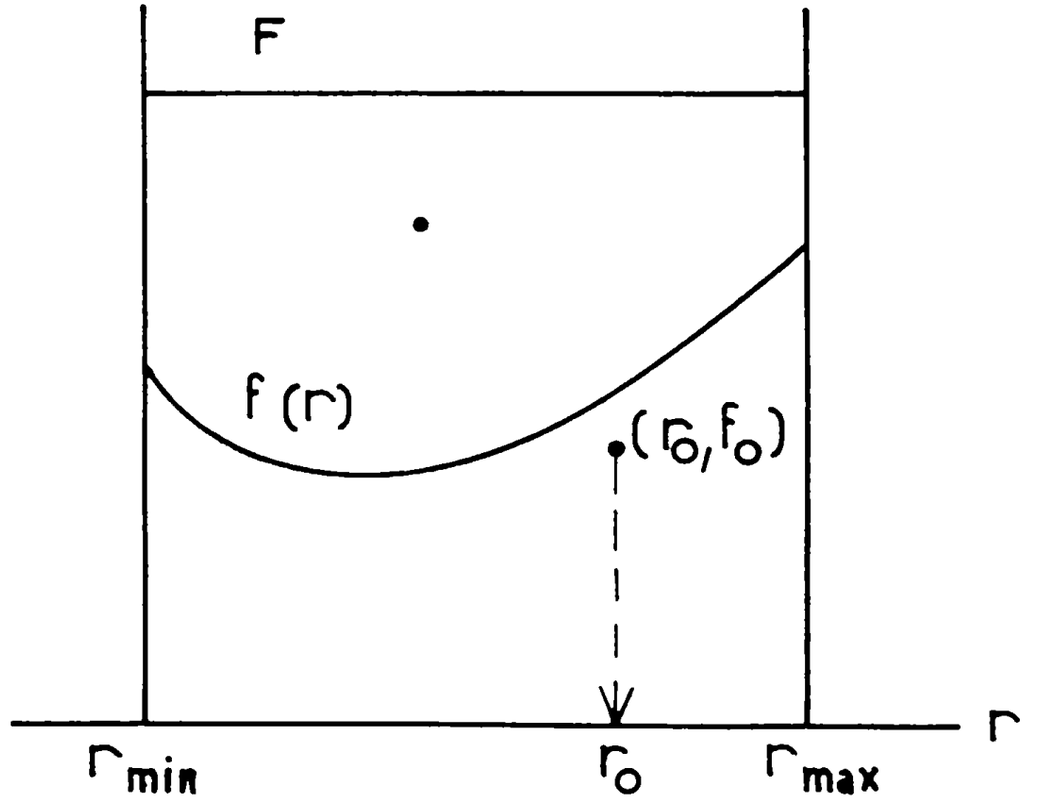


Fig. 2. The rejection technique.

- If random point (r_0, f_0) lies below $f(r)$, accept r_0 as new position
- Else reject and repeat

Agenda

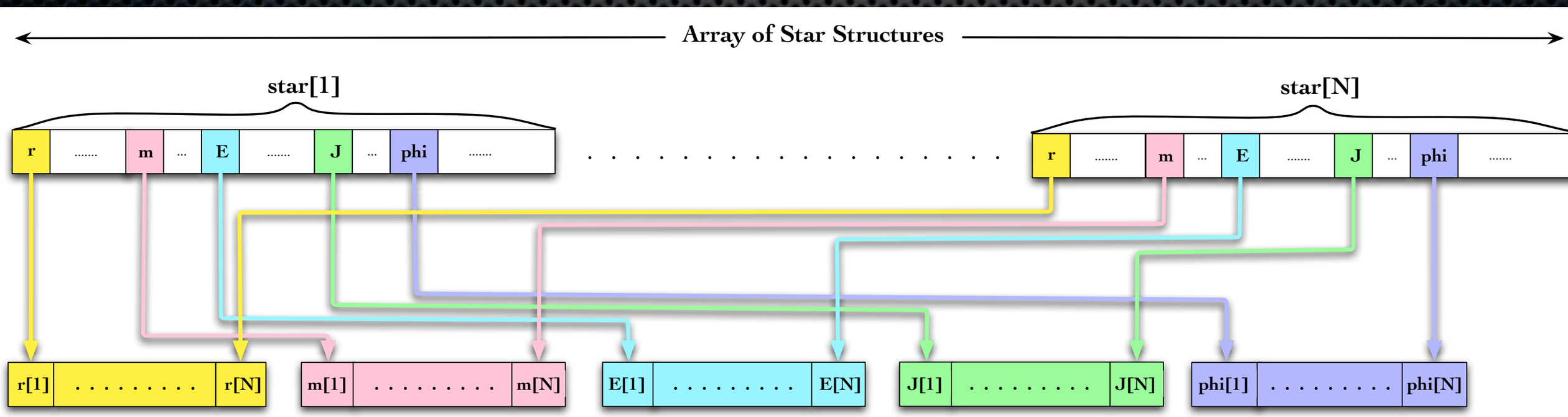
- Dense Stellar Systems
- The Monte Carlo Method
- Performance Bottlenecks
- **GPU Implementation**
- Results

GPU Implementation

- **One star, one thread**
  - Naturally suited for SIMT (Single Instruction, Multiple Thread) architecture
  - No data dependency between threads
  - Equal distribution of workload among threads

Memory Optimization

Original Data Structure



Radius Mass Energy Angular Momentum Potential

Reorganized Data Structure

Parallel Random Number Generation

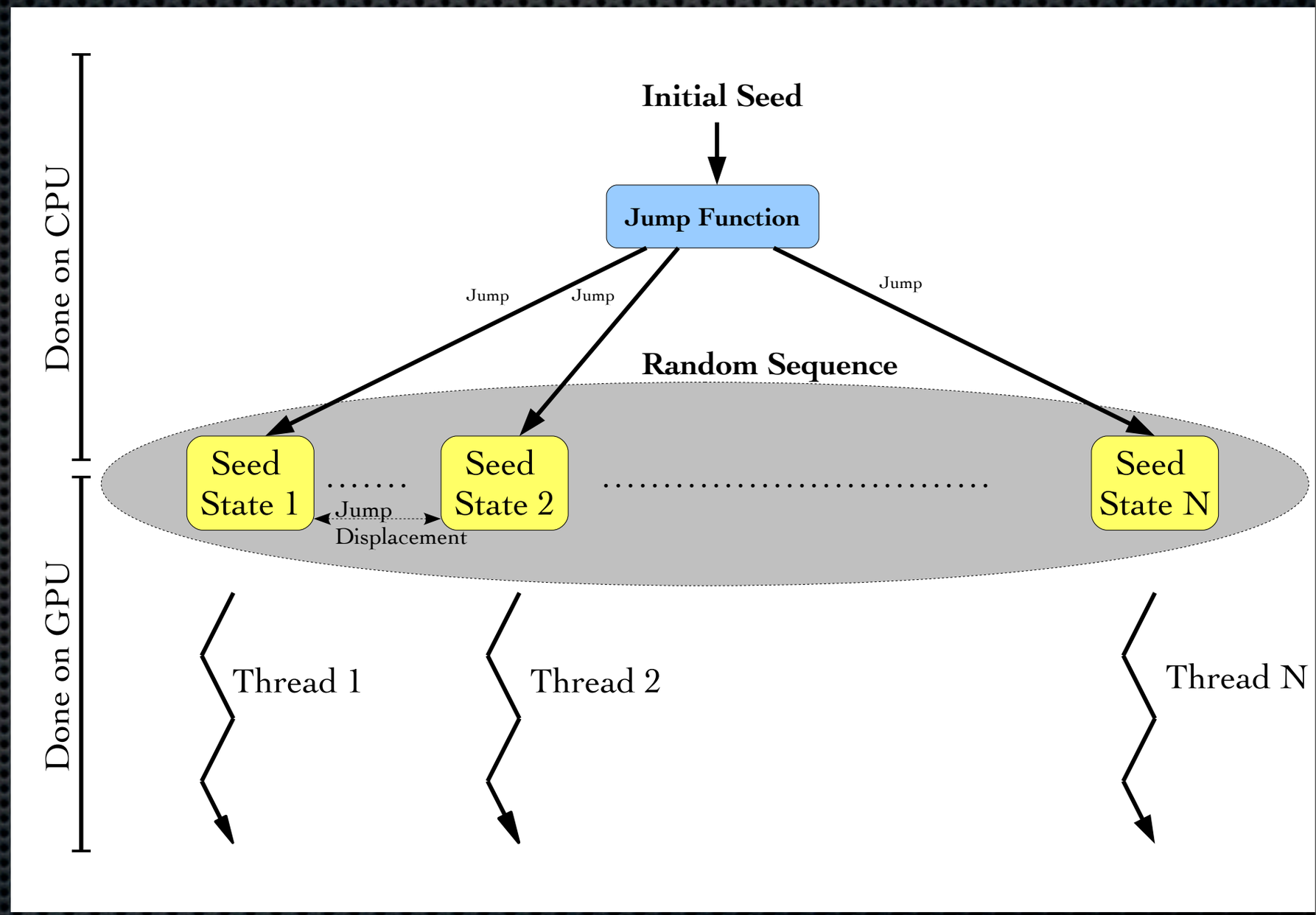
- Results of a parallel Monte Carlo code depends on
  - quality of RNG
  - parallel generation method
- RNG characteristics
  - state variables - used to calculate random numbers
  - period - number of calls after which the first number repeats
  - seeding - to initialize to a starting state in the sequence
- We use combined Tausworthe 113 - period of 2^{113}

Parallel Random Number Generation

- Parallel generation
  - naive method
    - each thread starts with its own copy of an RNG and its states
    - RNGs are initialized with different seeds
    - only assures different starting place in the same sequence, but sequences might eventually overlap
  - good method
    - should ensure statistical independence

Parallel Random Number Generation

- Split up single random sequence
- Naive algorithm - impractical
- Solution: **Jump Function**



J. C. Collins, "Testing, Selection, and Implementation of Random Number Generators," Army Research Laboratory, pp. 4, 41, 2008.

Agenda

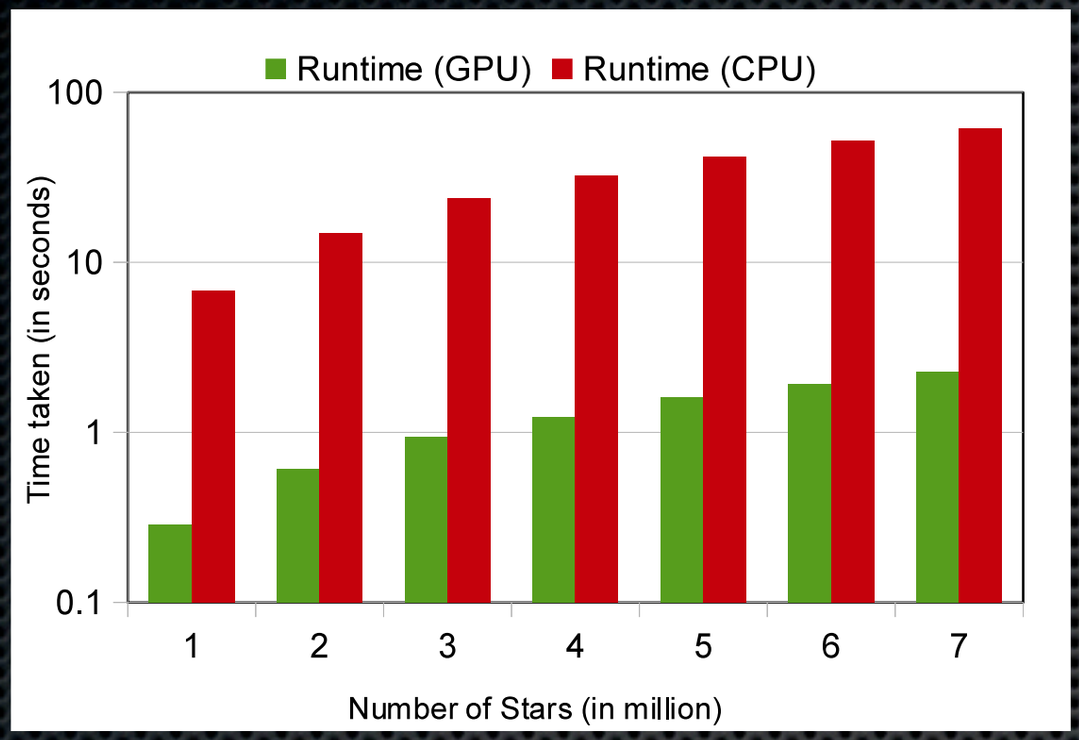
- Dense Stellar Systems
- The Monte Carlo Method
- Performance Bottlenecks
- GPU Implementation
- **Results**

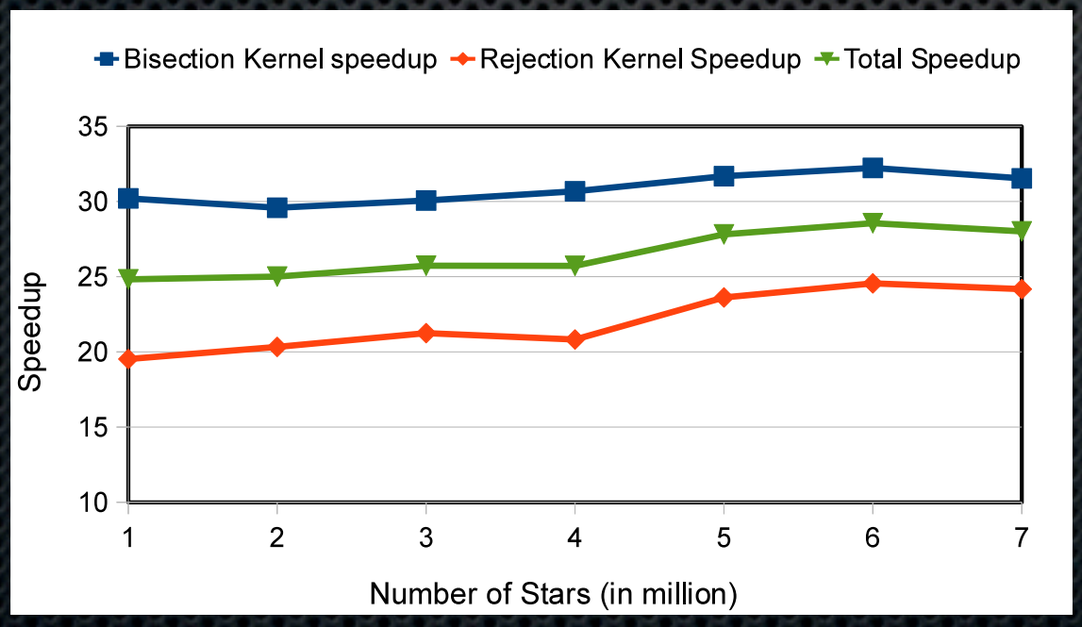
Performance Evaluation

- CPU
  - Single core of AMD Phenom™ Quad-Core
  - 8 GB Memory, 2.65 GHz
- GPU
  - NVIDIA GeForce GTX 280
  - 30 Multiprocessors (240 cores)
  - 1 GB Global Memory, 1.35 MHz

- Single mass cluster, Plummer density profile
- Number of stars: 1 million - 7 million

Results

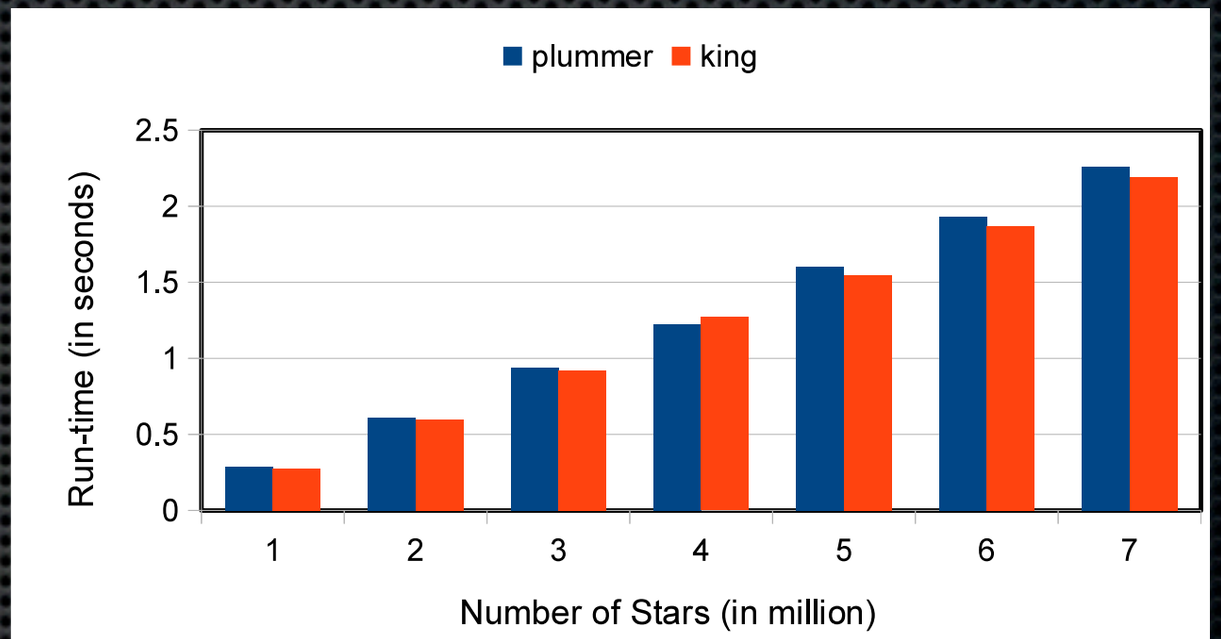


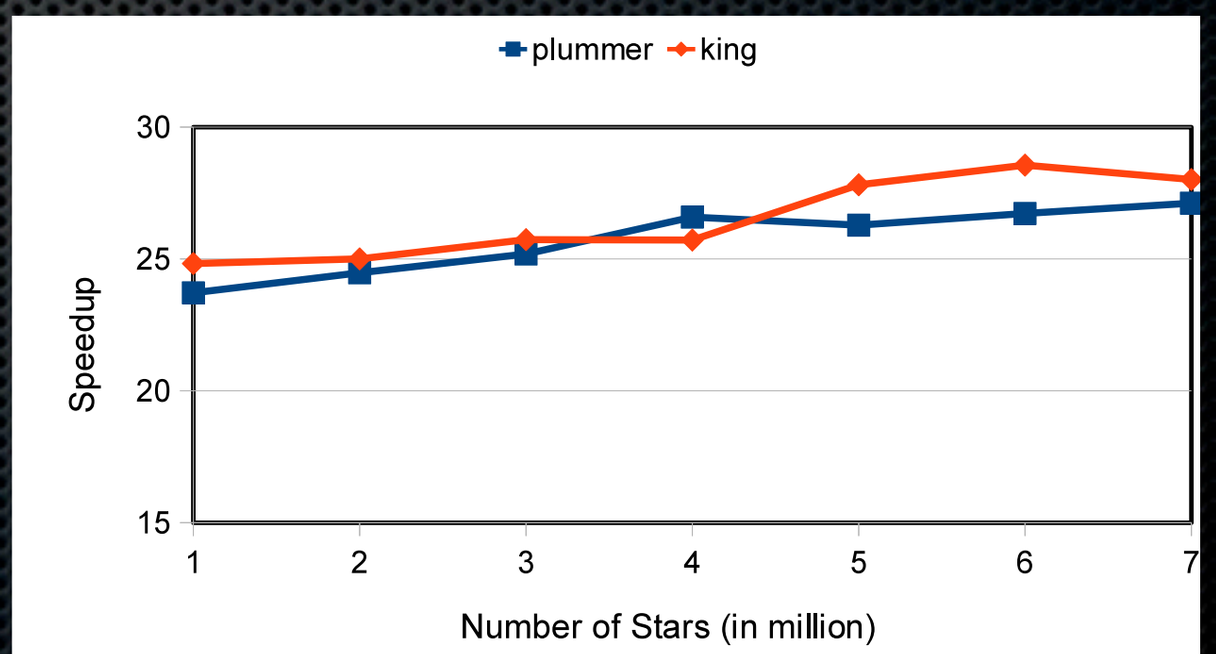


- All variables are in Double Precision
- GeForce GTX 280 has only 1 DPU/SM => 1/8th processing speed as that for Single Precision
- Maximum kernels Speedup: ~ 28X
- Overall code: ~ 2X

Results

- Effect of performance with change in physical cluster properties
- Tested by using clusters with Plummer and King density profiles
- No significant difference - implementation performs equally well on both configurations





Future Work

- Further optimizations
  - Current results do not include data transfer overhead - Interleaving with computation
  - Use of texture memory
  - Rerun experiments on newer Fermi GPU with L1/L2 cache

Future Work

- Target: **Galactic nuclei** simulations (**N ~ 10^9**)
- Fully parallel code - other parts not ideal for GPU due to high branch divergence, and huge data size
- Hybrid Parallelization: MPI + CUDA
- Large-scale clusters (CPUs + GPUs)