UNIVERSITEIT VAN AMSTERDAM

# N-Body Simulation

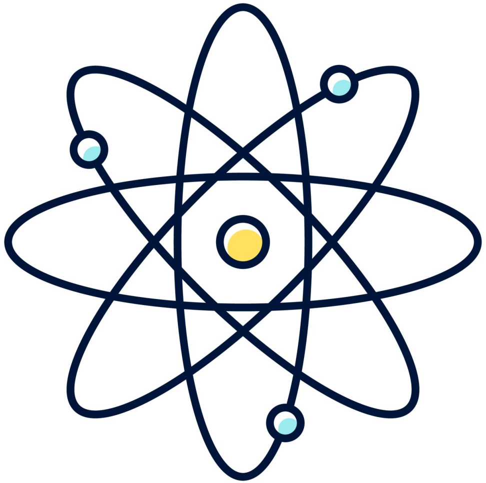
## Barnes-Hut Approximation

Foivos Papapanagiotakis-Bousy
Andre Palheiros da Silva
Tim van Kemenade
Dymitr Lubczyk

Course: Performance Engineering
Prof: Ana Varbanescu

# Introduction: N-Body Simulation

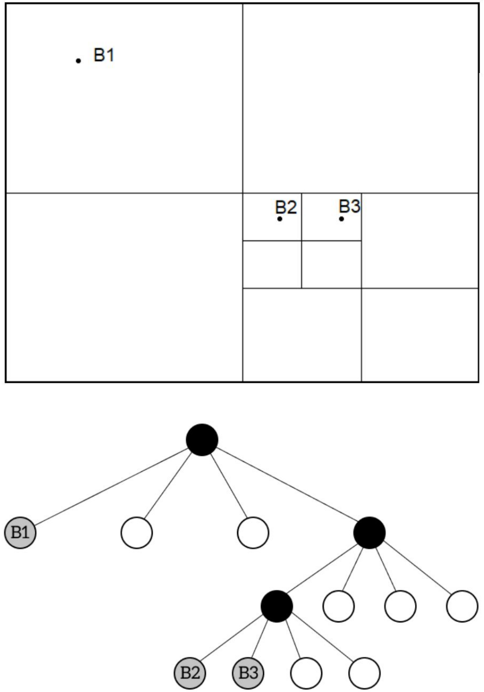
- Used to predict future particle (body) positions
- Each body has:
  - Mass
  - Position
  - Velocity
- The simulation runs for a given number of iterations
- Each iteration:
  - Body positions are updated based on velocities
  - Body velocities are updated by calculating the forces acting on the bodies using Newton's formulas
- By nature $O(n^2)$
- Goal: Optimize & Model for execution time



# Introduction: Barnes-Hut Approximation

- Estimated to be O(nlogn)
- Produces a quadtree based on the positions of bodies
- Approximates distant body forces by grouping distant bodies
  - Approximation occurs when: $\frac{diag}{dist} < \theta$
- Aiming for a maximum error of less than 1% after 50 iterations

| **Theta** | **Error** |
|---|---|
| 0.01 | 0% |
| 0.05 | 0.06% |
| 0.1 | 0.13% |
| 0.5 | 1.05% |
| 1 | 3.94% |
| 1.5 | 11.74% |



# Experimental Setup

## Software

**Universe configuration**

| | |
|---|---|
| **Bodies** (32-bit unsigned int) | [1e2, 1e4] |
| **Mass** [kg] (64-bit float) | [1e10, 1e40] |
| **Velocity-X** [m] (64-bit float) | [−1e6, 1e6] |
| **Velocity-Y** [m] (64-bit float) | [−1e6, 1e6] |
| **Position-X** [m/s] (64-bit float) | [−5e16, 5e16] |
| **Position-Y** [m/s] (64-bit float) | [−5e16, 5e16] |

**Simulation configuration**

| | |
|---|---|
| **Iterations** (32-bit unsigned int) | 50 |
| **Iteration Duration** [s] (64-bit float) | 3600 |
| **Theta** (Barnes-Hut only) (64-bit float) | 0.5 |

## Hardware

**Intel® Xeon® E5-2630 v3 Processor**

| | |
|---|---|
| Physical cores | 8 |
| Logical cores | 16 |
| Base frequency | 2.40 GHz |
| Turbo frequency | 3.20 GHz |
| Size L1 cache | 32 KB |
| Size L2 cache | 256 KB |
| Size L3 cache | 20 MB |
| Latency L1 cache | 4 cycles |
| Latency L2 cache | 12 cycles |
| Latency L3 cache | 34 cycles |
| Cache line size | 64 Bytes |

# Prototype 1: Model

| Symbol | Description |
|---|---|
| $t_{pu}$ | Time required to update the position of a single body |
| $t_{tl}$ | Time required to insert a body in a quads subtree |
| $t_q$ | Time required to generate empty subtree |
| $t_{rc}$ | Time required to call a function |
| $t_{fc}$ | Time required to calculate a force acting on a body |
| $t_{dc}$ | Time required to calculate the Euclidean distance between any two points |
| $Dp(n)$ | Estimator of average tree depth |
| $Fc(n)$ | Estimator of number of force calculations per body |
| $Nv(n)$ | Estimator of number of node visits per body |
| $Lvf(n)$ | Estimator of ratio: leaf_visits / total_visits |

$$T = T_{pu} + T_{tg} + T_{vu}$$

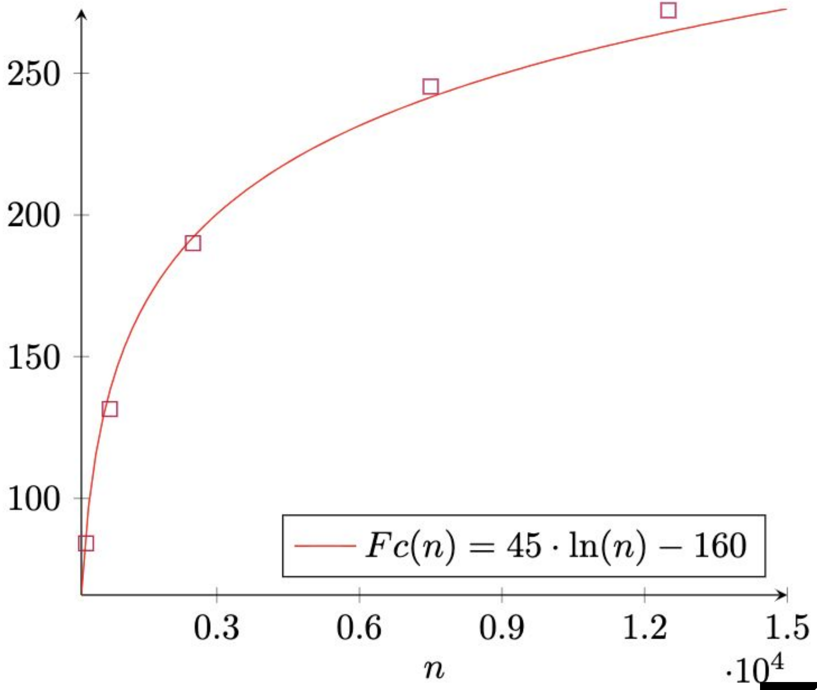
$$T_{pu} = N \cdot t_{pu}$$

$$T_{tg} = N \cdot \left( Dp(N) \cdot t_{tl} + \frac{4(t_q + t_{rc})}{3} \right)$$

$$T_{vu} = N \cdot \left( Fc(N) \cdot t_{fc} + Nv(N) \cdot t_{rc} + (1 - Lvf(N)) \cdot Nv(N) \cdot t_{dc} \right)$$

# Prototype 1: Approximation Functions

- Understand the nature of the phenomena
- Gather measurement on a training set
- Train the model
- Validate it on a test set
- Enjoy your approximation



# Prototype 1: Model Calibration

| **Symbol** | **Value** |
|---|---|
| $t_{pu}$ | 5ns |
| $t_{tl}$ | 29ns |
| $t_q$ | 60ns |
| $t_{rc}$ | 0.6ns |
| $t_{fc}$ | 55ns |
| $t_{dc}$ | 34ns |
| $Dp(n)$ | $1.64 \cdot \ln(n) - 0.91$ |
| $Fc(n)$ | $45 \cdot \ln(n) - 160$ |
| $Nv(n)$ | $65 \cdot \ln(n) - 197$ |
| $Lvf(n)$ | $-0.0879 \cdot \ln(n) + 1.13$ |

$$T = N \cdot (194.25 \cdot \ln^2(N) + 1,685.50 \cdot \ln(N) - 7,988.05)$$

# Prototype 1: Results

| N | Measured[s] | Predicted[s] | Error |
|---|---|---|---|
| 10000 | 0.2995760 | 0.2403889 | 19.76% |
| 5000 | 0.1132730 | 0.1024209 | 9.58% |
| 1000 | 0.01404501 | 0.0129514 | 7.79% |
| 500 | 0.00545305 | 0.0050089 | 8.15% |
| 100 | 0.00033501 | 0.0003927 | 17.22% |

| N | $\mathbf{T_{pu}}$ | $\mathbf{T_{tg}}$ | $\mathbf{T_{vu}}$ |
|---|---|---|---|
| 10000 | 0.01% | 1.77% | 98.23% |
| 5000 | 0.01% | 2.17% | 97.82% |
| 1000 | 0.02% | 3.41% | 96.57% |
| 500 | 0.03% | 4.42% | 95.55% |
| 100 | 0.15% | 15.10% | 84.76% |

# Prototype 2: Optimisations

- Square root and division computations are expensive:
  - $\frac{diag}{dist} < \theta \longrightarrow diag^2 < \theta^2 \cdot dist^2$
- `#pragma omp parallel for schedule(dynamic)`
  - Applied on the *Velocity Update* loop
  - Dynamic scheduling determined optimal due to load imbalance

# Prototype 2: Model

$$T = T_{pu} + T_{tg} + T_{vu}$$

$$T_{pu} = N \cdot t_{pu}$$

$$T_{tg} = N \cdot \left( Dp(N) \cdot t_{tl} + \frac{4(t_q + t_{rc})}{3} \right)$$

$$T_{vu} = \frac{N}{P} \cdot \Big( Fc(N) \cdot Tfc(P) + Nv(N) \cdot t_{rc} + (1 - Lvf(N)) \cdot Nv(N) \cdot Tdc(P) \Big)$$

| **Symbol** | **Description** |
|---|---|
| $Tfc(p)$ | Estimator of time required to calculate a force acting on a body |
| $Tdc(p)$ | Estimator of time required to calculate the Euclidean distance between any two points |

# Prototype 2: Model Calibration

| **Symbol** | **Value** |
|---|---|
| $t_{pu}$ | 8ns |
| $t_{tl}$ | 68ns |
| $t_q$ | 105ns |
| $t_{rc}$ | 0.6ns |
| $Tfc(p)$ | $2.23 \cdot p + 52.1$ |
| $Tdc(p)$ | $2.27 \cdot p + 28.5$ |
| $Dp(n)$ | $1.64 \cdot \ln(n) - 0.91$ |
| $Fc(n)$ | $45 \cdot \ln(n) - 160$ |
| $Nv(n)$ | $65 \cdot \ln(n) - 197$ |
| $Lvf(n)$ | $-0.0879 \cdot \ln(n) + 1.13$ |

$$T = N \cdot (12.96 \cdot \ln^2(N) + 153.38 \cdot \ln(N) - 211.74 + \frac{162.83 \cdot \ln^2(N) + 1649.16 \cdot \ln(N) - 7724.31}{P})$$

# Prototype 2: Results

| N | P | Measured[s] | Predicted[s] | Error |
|---|---|---|---|---|
| 10000 | 2 | 0.104973 | 0.129726 | 23.58% |
| 10000 | 4 | 0.066044 | 0.076354 | 15.61% |
| 10000 | 8 | 0.040812 | 0.049668 | 21.70% |
| 1000 | 2 | 0.007450 | 0.007214 | 3.16% |
| 1000 | 4 | 0.004567 | 0.004339 | 4.98% |
| 1000 | 8 | 0.003095 | 0.002902 | 6.23% |
| 100 | 2 | 0.000495 | 0.000246 | 50.35% |
| 100 | 4 | 0.000364 | 0.000161 | 55.64% |
| 100 | 8 | 0.000343 | 0.000119 | 65.29% |

| N | P | $T_{pu}$ | $T_{tg}$ | $T_{vu}$ |
|---|---|---|---|---|
| 10000 | 2 | 0.07% | 10.49% | 89.44% |
| 10000 | 4 | 0.12% | 17.04% | 82.85% |
| 10000 | 8 | 0.20% | 28.26% | 71.54% |
| 1000 | 2 | 0.14% | 14.94% | 84.92% |
| 1000 | 4 | 0.23% | 24.21% | 75.55% |
| 1000 | 8 | 0.38% | 38.10% | 61.52% |
| 100 | 2 | 0.28% | 21.32% | 78.40% |
| 100 | 4 | 0.37% | 29.49% | 70.14% |
| 100 | 8 | 0.31% | 32.78% | 66.91% |

# Prototype 3: Optimisations

- Bulk quads initialisation
  - Less system calls
  - Better caching as quadtree is stored in an array
- Replacing **std::list** with **std::vector**
  - Better caching behaviour, faster iterations
  - Allocated with initial size of 40

# Prototype 3: Model

$$T = T_{pu} + T_{tg} + T_{vu}$$

$$T_{pu} = N \cdot t_{pu}$$

$$T_{tg} = N \cdot \Big( Dp(N) \cdot t_{tl} + 4(t_q + t_{rc}) \Big)$$

$$T_{vu} = \frac{N}{P} \cdot \Big( Fc(N) \cdot Tfc(P) + Nv(N) \cdot t_{rc} + (1 - Lvf(N)) \cdot Nv(N) \cdot Tdc(P) \Big)$$

# Prototype 3: Model Calibration

| **Symbol** | **Value** |
|---|---|
| $t_{pu}$ | 8ns |
| $t_{tl}$ | 11ns |
| $t_q$ | 80ns |
| $t_{rc}$ | 0.6ns |
| $Tfc(p)$ | $2.16 \cdot p + 35.6$ |
| $Tdc(p)$ | $2.27 \cdot p + 28.5$ |
| $Dp(n)$ | $1.64 \cdot \ln(n) - 0.91$ |
| $Fc(n)$ | $45 \cdot \ln(n) - 160$ |
| $Nv(n)$ | $65 \cdot \ln(n) - 197$ |
| $Lvf(n)$ | $-0.0879 \cdot \ln(n) + 1.13$ |

$$T = N \cdot (12.96 \cdot \ln^2(N) + 56.75 \cdot \ln(N) + 32.92 + \frac{162.83 \cdot \ln^2(N) + 947.16 \cdot \ln(N) - 5228.31}{P})$$

# Prototype 3: Results

| N | P | Measured[s] | Predicted[s] | Error |
|---|---|---|---|---|
| 10000 | 2 | 0.090472 | 0.102891 | 13.73% |
| 10000 | 4 | 0.049679 | 0.059738 | 20.25% |
| 10000 | 8 | 0.030900 | 0.038161 | 23.50% |
| 1000 | 2 | 0.006213 | 0.005581 | 10.18% |
| 1000 | 4 | 0.003907 | 0.003313 | 15.20% |
| 1000 | 8 | 0.002480 | 0.002180 | 12.11% |
| 100 | 2 | 0.000325 | 0.000188 | 42.31% |
| 100 | 4 | 0.000284 | 0.000122 | 56.91% |
| 100 | 8 | 0.000250 | 0.000090 | 64.12% |

| N | P | Tpu | Ttg | Tvu |
|---|---|---|---|---|
| 10000 | 2 | 0.09% | 5.19% | 94.72% |
| 10000 | 4 | 0.17% | 8.88% | 90.95% |
| 10000 | 8 | 0.28% | 14.30% | 85.43% |
| 1000 | 2 | 0.16% | 9.20% | 90.64% |
| 1000 | 4 | 0.28% | 14.37% | 85.35% |
| 1000 | 8 | 0.47% | 22.76% | 76.77% |
| 100 | 2 | 0.44% | 21.91% | 77.65% |
| 100 | 4 | 0.49% | 26.52% | 72.99% |
| 100 | 8 | 0.59% | 28.64% | 70.76% |

# Calibrated Models: Overview

**Model 1:**

$$T = N \cdot (194.25 \cdot \ln^2(N) + 1,685.50 \cdot \ln(N) - 7,988.05)$$

**Model 2:**

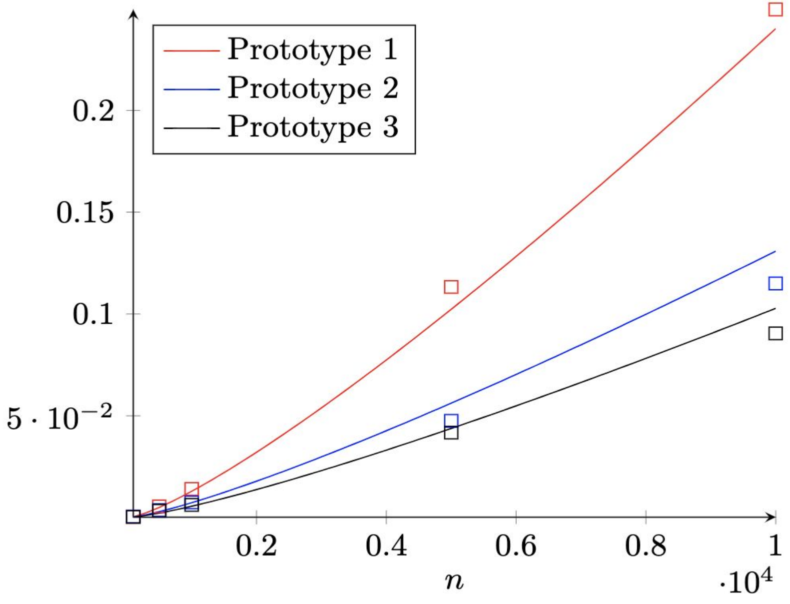
$$T = N \cdot (12.96 \cdot \ln^2(N) + 153.38 \cdot \ln(N) - 211.74 + \frac{162.83 \cdot \ln^2(N) + 1649.16 \cdot \ln(N) - 7724.31}{P})$$

**Model 3:**

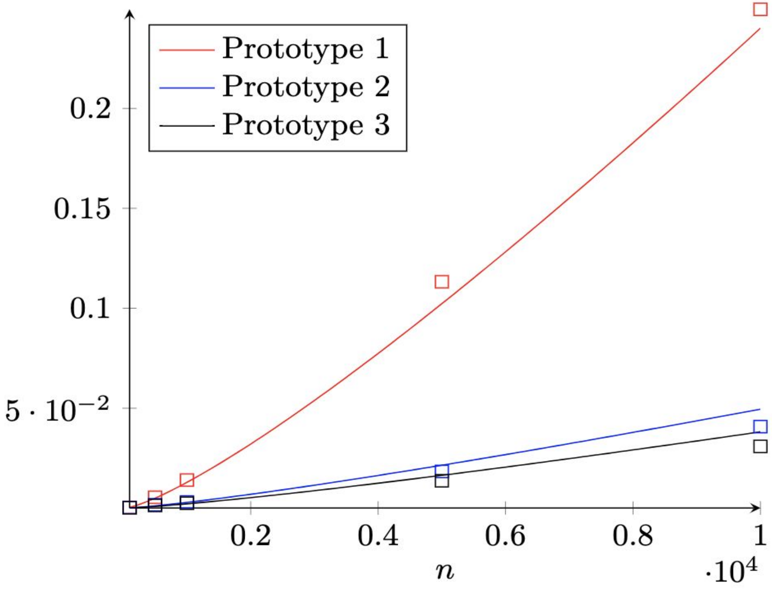
$$T = N \cdot (12.96 \cdot \ln^2(N) + 56.75 \cdot \ln(N) + 32.92 + \frac{162.83 \cdot \ln^2(N) + 947.16 \cdot \ln(N) - 5228.31}{P})$$

# Performance Prediction

**2 threads**

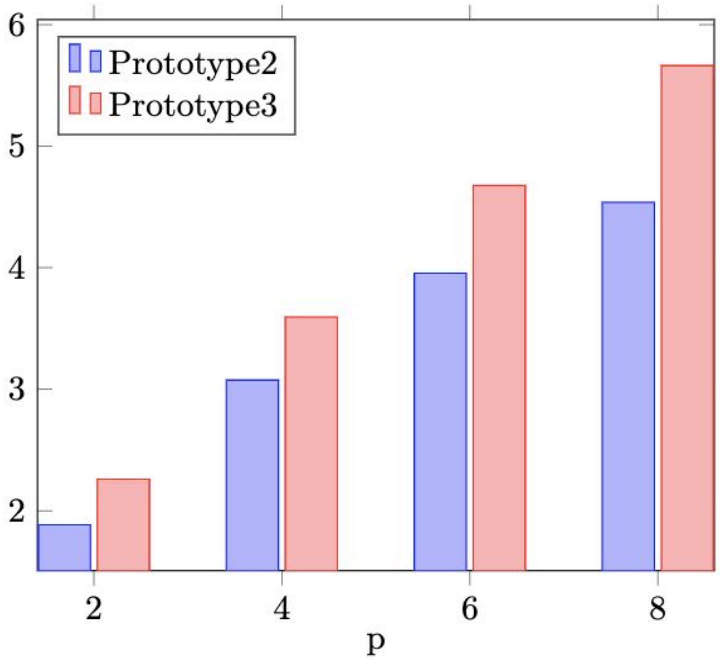



**8 threads**

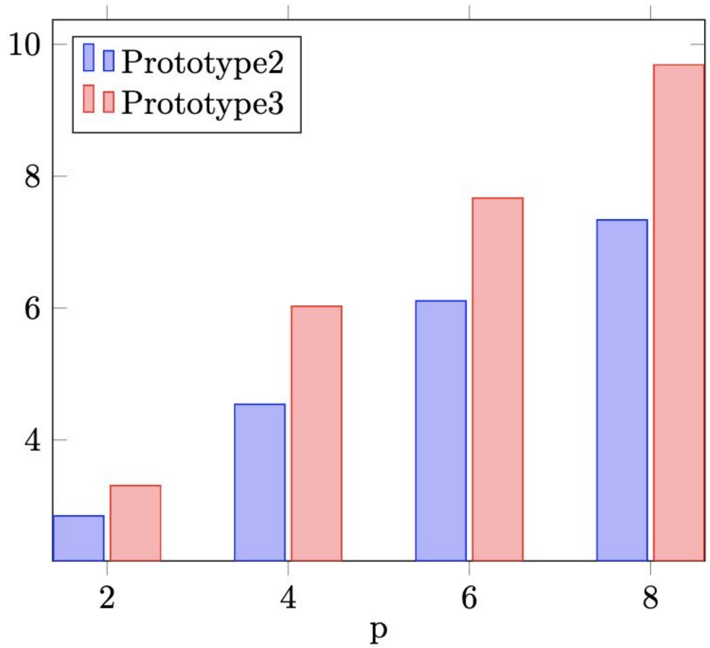


# Performance Results

**1k particles**




**10k particles**



# Final Result

**1.89s -> 0.0376s**

**Speedup: 50.4**