Эффективное использование гибридных архитектур вида CPU+GPU в библиотеке *ttgLib*

Суперкомпьютеры *vs* Обычные компьютеры с GPU

(попытка сравнения их общих характеристик)

Система	Суперкомпьютер СФУ (Красноярск)	Суперкомпьютер УГАТУ (Уфа)	СКИФ-МГУ (Москва)	Компьютер из стандартных комплектующих #1 [1]	Компьютер из стандартных комплектующих #2 [2]	Компьютер из стандартных комплектующих #3 [3]	Asus ESC-1000 [4]
Пиковая производительность	16.8 TFlops	19.8 TFlops	60 TFlops	1.5 TFlops	5.1 TFlops	6.1 TFlops	3.8 TFlops
Достижимая производительность	13 TFlops	15.3 TFlops	47 TFlops	-	-	-	1.1 TFlops
Количество ядер	1808	2128	5000	336 (GPU) 4 (CPU)	1200 (GPU) 4 (CPU)	1440 (GPU) 4 (CPU)	960 (GPU) 4 (CPU)
Объём памяти	902 GB	2 150 GB	5 500 GB	16 GB (CPU) 3 GB (GPU)	8 GB (CPU) 4.3 GB (GPU)	24 GB (CPU) 12 GB (GPU)	24 GB (CPU) 16 GB (GPU)
Дисковая подсистема	4 TB	26.7 TB	15 TB	0.5 TB	1 TB	2 TB	0.5 TB
Занимаемое место	-	6 шкафов	96 м^2	Под столом	Под столом	Под столом	Под столом
Энергопотребление	-	85 KW	330 KW	0.8 KW	1.2 KW	1.5 KW	1.1 KW
Стоимость	83 млн. руб	130 млн. руб	231 млн. руб	~40 000 руб	~80 000 руб	~160 000 руб	~420 000 руб
Цена за GFlops	**4 940 рублей**	**6 565 рублей**	**3 850 рублей**	**27 рублей**	**16 рублей**	**26 рублей**	**1 100 рублей**

(1) MB ASUS Striker II + Intel Core 2 Quad Q9650 + 3 * NVidia GeForce 9800GT + 4 * 4 GB DDR2.
(2) MB ASUS P7P55 WS + Intel Core i7 860 + 5 * NVidia GeForce 275 + 4 * 2 GB DDR3.
(3) MB ASUS P6T6 WS + Intel Core i7 960 + 6 * NVidia GeForce 285 + 6 * 4 GB DDR3.
(4) http://www.3dnews.ru/news/asus_esc_1000_nastolnii_superkomputer_s_1_1_tflops/

Недостатки суперкомпьютеров:

- Лимитирование доступа по времени и ресурсам
  На кластере СКИФ-МГУ на решение задачи выделяют не более 20 000 процессоро-часов.
- Сложность получения логина
  Может потребоваться написание *K* заявлений, а также получение *N* ключей и *M* паролей
- Отсутствие общепринятых стандартов
  Каждый суперкомпьютер является уникальным (как программно, так и аппаратно), что может затруднить переход на другие системы

Недостатки графических процессоров:

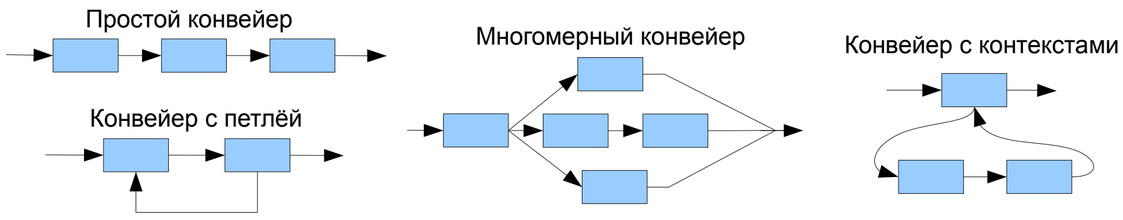
- Не все задачи имеют решение
  Специфичность архитектуры делает сложным эффективное решение многих задач (например, не имеющих явного параллелизма по данным)
- Проблемы совместного использования множества GPU
  В большинстве API решение проблем нахождения устройств и распределения вычислительной нагрузки между ними ложится на пользователя

О библиотеке *ttgLib*

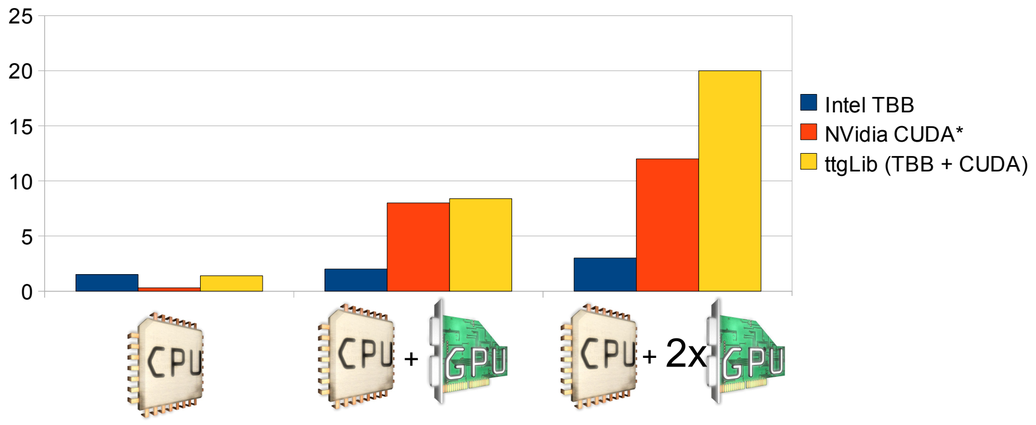
ttgLib — библиотека для быстрой разработки вычислительноёмких приложений, позволяющая интегрировать различные технологии и одновременно использовать все доступные вычислительные устройства (CPU + GPU).

Отличительные особенности

Гибкая потоко-ориентированная модель. Основой библиотеки *ttgLib* является параллельный примитив ttg::pipeline, позволяющий элегантно решить практически любую задачу. В нашем понимании конвейер может быть многомерным, поддерживать петли и динамическую переупаковку потоков данных, а также предоставлять возможности для определения контекстов, предотвращающих перемешивание различных данных. Одной из отличительных его особенностей является поддержка рекурсивных потоков данных и внутренних событий, что позволяет решить даже задачи с рекурсией.



Поддержка вычислителей с различными архитектурами. Каждое звено конвейера (единица, производящая обработку данных) может содержать дополнительные методы для альтернативной обработки данных с использованием конкретного вычислительного устройства. Это позволяет системе времени выполнения динамически загружать все доступные вычислители, а также обеспечивать работоспособность в случае отсутствия требуемых вычислительных устройств или их несовместимости с требуемым API.



Интеграция со внешними утилитами. Во время работы приложения, использующего библиотеку *ttgLib*, к нему можно подсоединить внешние утилиты, каждая из которых позволяет просматривать и даже изменять некоторые параметры «незаметно» для основной программы. Это, с одной стороны, позволяет существенно сократить код, необходимый для решения требуемой задачи, а с другой, делает возможным провести более гибкую настройку под конкретную систему.

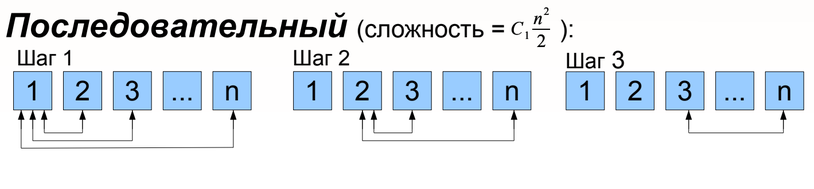
Так, например, пользователь может не только посмотреть на загрузку используемых устройств и установить для них приоритеты, но и может поменять значение параметров программы или даже перестроить конвейер.

Модельная задача N тел

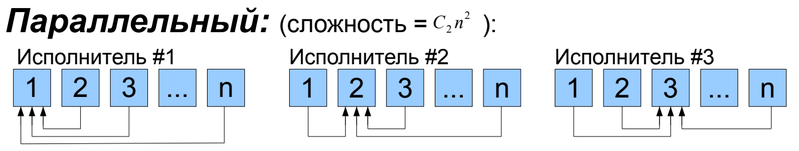
Требуется провести моделирование системы из N тел, каждое из которых обладает ненулевой массой и притягивает другие тела данной системы.

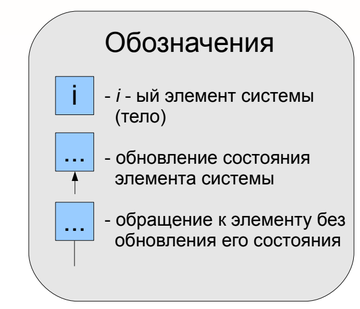
Используемые алгоритмы

- ***Последовательный*** (сложность = $C_1\frac{n^2}{2}$):

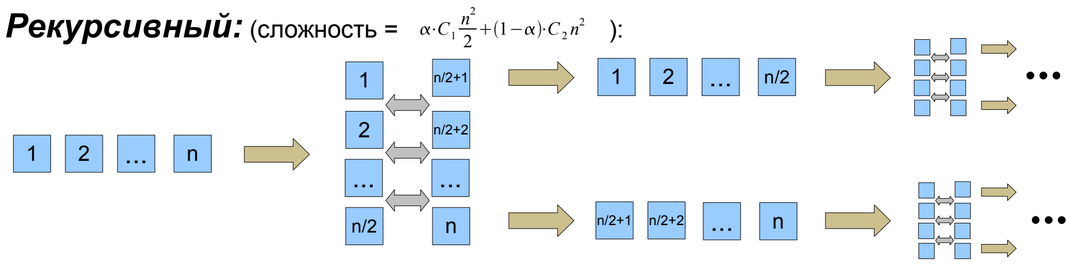


- ***Параллельный:*** (сложность = $C_2 n^2$):





- ***Рекурсивный:*** (сложность = $\alpha \cdot C_1\frac{n^2}{2} + (1-\alpha)\cdot C_2 n^2$):

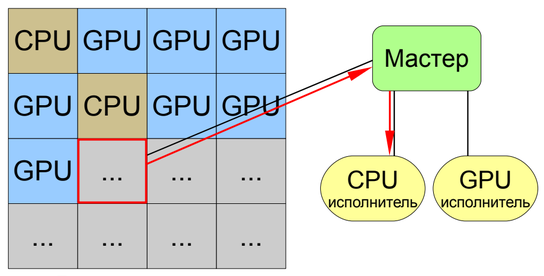


Проблема балансировки нагрузки

Какую часть тел обрабатывать на конкретном вычислительном устройстве?

Динамическая балансировка

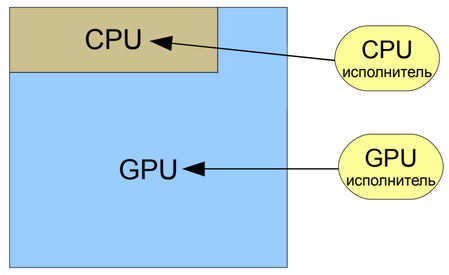
Паттерн «Мастер-Исполнители»



Достоинства:
- Легко реализовать

Недостатки:
- Не эффективен при $GPU_{perf} >> CPU_{perf}$
- Большие накладные расходы

Статическая балансировка

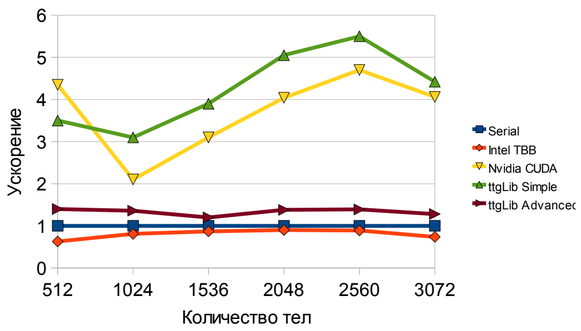


Достоинства:
- Достигается оптимальное распределение

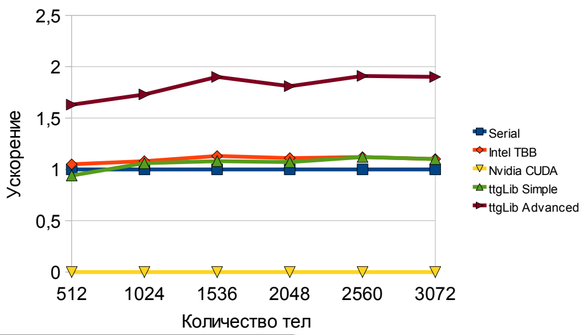
Недостатки:
- Требуется информация о производительности вычислителей

Сравнение реализаций

CPU: Intel Atom 270 ***GPU:*** GeForce 9300



CPU: Intel Pentium D 945



Обозначения

Serial — реализация последовательного алгоритма
Intel TBB — реализация параллельного алгоритма с использованием библиотеки Intel TBB
NVidia CUDA — реализация параллельного алгоритма с использованием библиотеки NVidia CUDA
ttgLib Simple — реализация параллельного алгоритма для связки CPU + GPU
ttgLib Advanced — реализация рекурсивного алгоритма для связки CPU+GPU