Разработка параллельных алгоритмов расчёта показателей надёжности сетей для исполнения на гибридных ВС

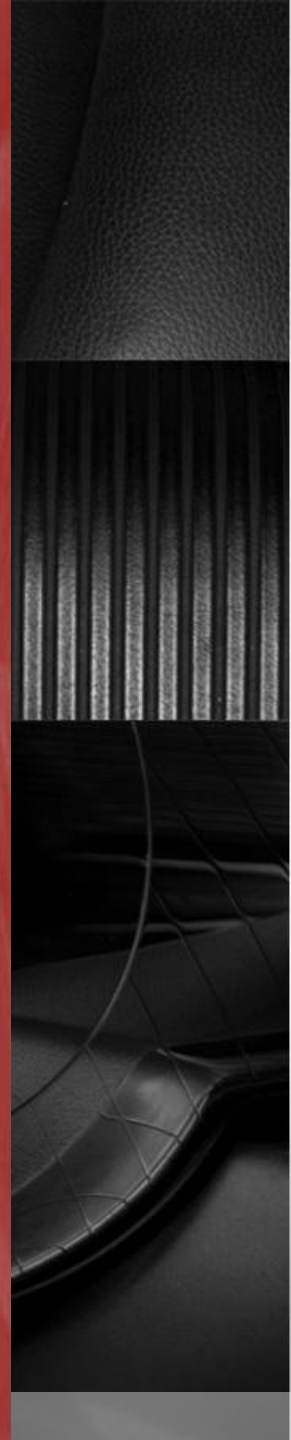
Студент 4 курса ФИТ

М.И. Никулин

Научный руководитель, д.т.н.,

Зав. лаб. МоДПрИС ИВМиМГ СО РАН

А.С. Родионов



Проблематика

- Планирование абстрактной связной сети становится трудоёмкой задачей, если оно включает в себя полный расчет связности, которая входит в класс NP-трудных задач.

Цель

- Исследование вопросов связанных с разработкой точных и приближённых алгоритмов расчёта показателей надёжности сетей с ненадёжными элементами, предназначенных для параллельной реализации на гибридных кластерах, содержащих CPU и GPU.

Задачи

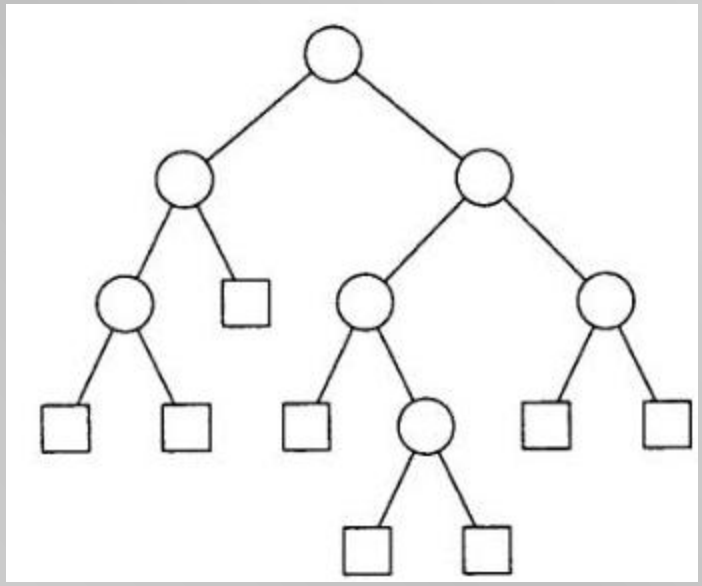
- Изучение последовательных алгоритмов вычисления средней вероятности парной связности в сетях с ненадёжными соединениями (граф с ненадёжными рёбрами).
- Разработка кумулятивного алгоритма принятия решения о надёжности сети по указанному критерию.
- Разработка параллельных версий алгоритмов.
- Реализация алгоритмов и проведение численных экспериментов для оценки их эффективности и масштабируемости.

Модель

- Случайный G(X, U) (n, m) – граф.
- Вершины абсолютно надежны. Ребра надежны с одинаковой вероятностью 0 <= p <= 1.
- R(G) = Pj R (G/j) + (1 – Pj) R(G\j), j из U (метод факторизации).

Точный алгоритм

- Строится бинарное дерево разбиения графа.
- Каждый переход – удаление или стягивание ребра.
- Листья – граф, для которого существует простая формула расчета связности.
- Последовательный алгоритм – поиск в глубину.



Параллелизация точного алгоритма

- Используется CUDA.
- Каждый переход по бинарному дереву – передача новой задачи незадействованному ядру.
- В случае получения ядром задачи-листа рассчитывается связность и начинается обратный ход.

Архитектурные проблемы

- Особенности архитектуры CUDA, в частности отсутствие взаимодействия между ядрами, не позволили реализовать алгоритм расчета надежности сети при помощи метода ветвления.
- Существует возможность организовать взаимодействие между ядрами при помощи разделяемой памяти, однако такой подход будет медленным и трудно реализуемым.
- Было принято решение провести эксперимент подсчета надежности сети полным перебором.

Полный перебор

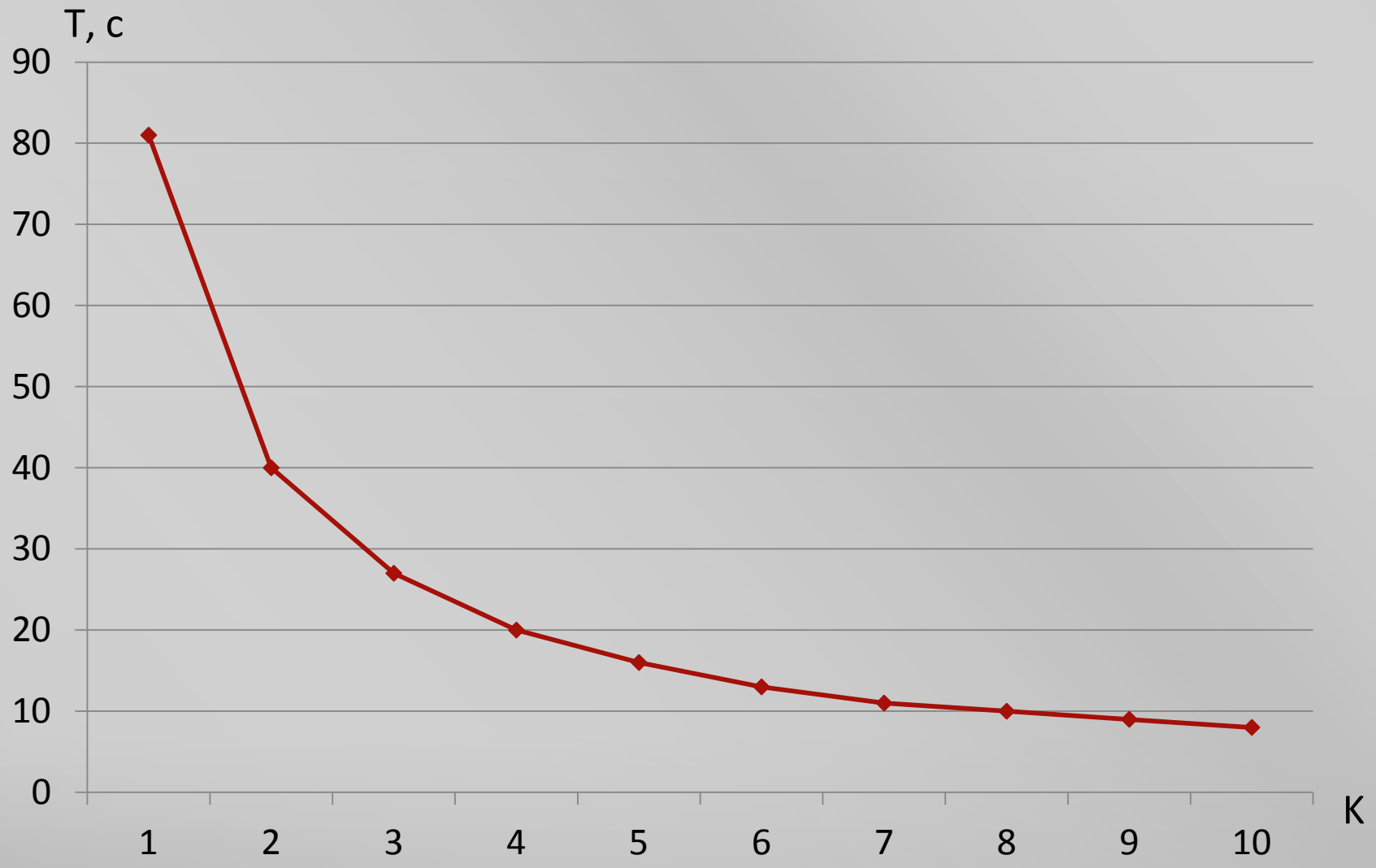
- За основу взята формула

$$N(G) = \sum_{i=0}^{m} p^i (1-p)^{m-i} \sum_{j=1}^{C_m^i} N^*(G_{ij})$$

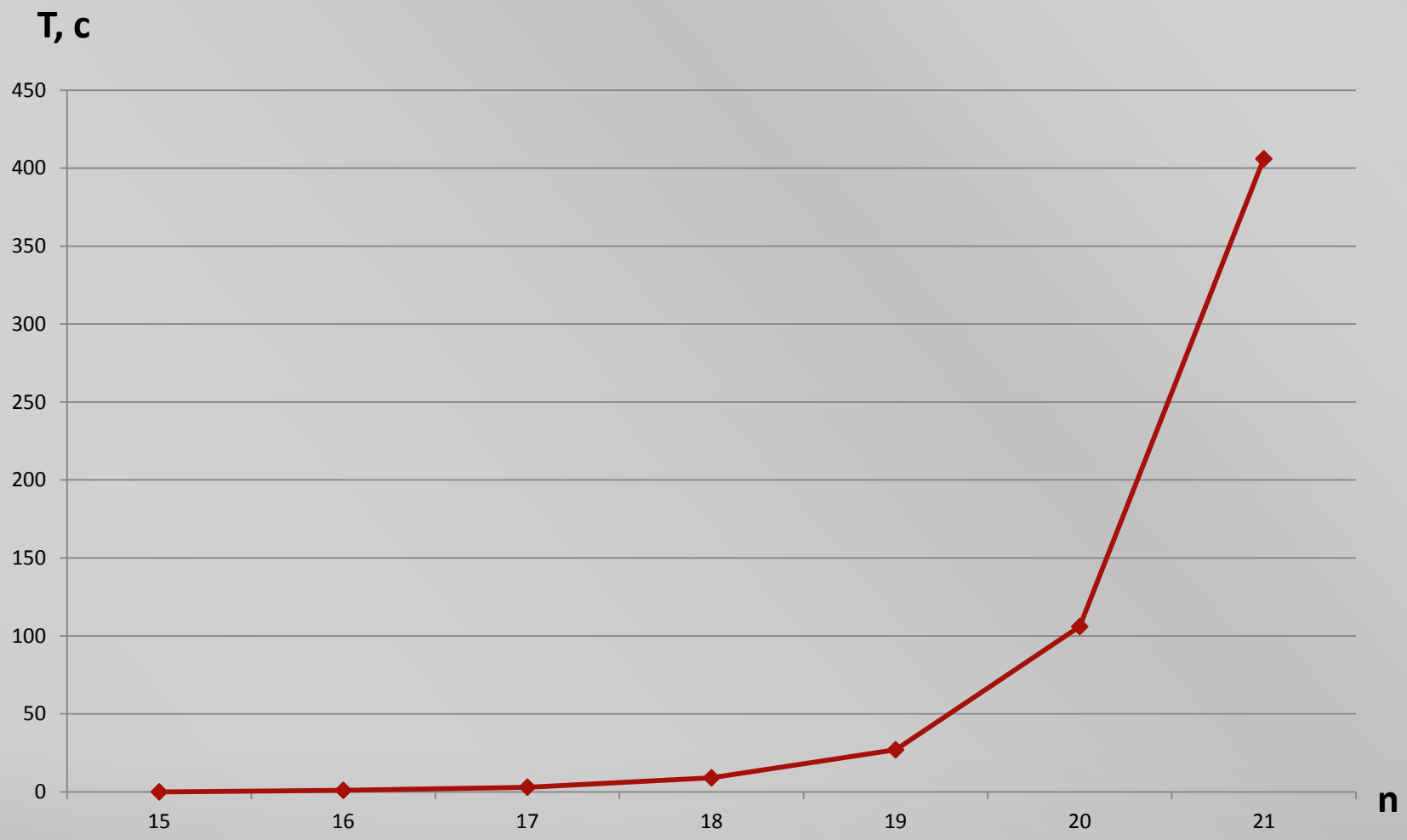
- Где Gij – j-ый вариант удаления из графа G ровно i рёбер (осуществляется с вероятностью p^i(1-p)^(m-i), N*(Gij) – число несвязных пар вершин в Gij.
- На Cuda первая сумма распараллеливается по Grid, вторая по Blocks.

Численный эксперимент

- Устройство GeForce GT 630m, 2 сопроцессора по 96 Cuda Cores, производительность 258 GFlops.
- При экспериментах определено, что скорость расчётов больше зависит от количества рёбер в графе, чем от количества вершин). При добавлении одного ребра к графу, время расчёта увеличивается в 2 и более раз.
- При равном количестве рёбер и разном количестве вершин (8, 10, 12, 15) время расчётов отличается на 15-20%.



Зависимость времени расчёта от количества ядер



Зависимость времени расчётов от количества рёбер графа

Средства разработки

- Microsoft Visual Studio 2010 C++ - стандартная среда разработки на языке C/C++.
- Nvidia Nsight Visual Studio Edition 3.0 – отладчик и анализатор кода.
- CUDA SDK 5.0.

Результаты

- Описаны причины сложности реализации и нецелесообразность использования метода факторизации на архитектуре CUDA.
- Показана возможность расчёта показателей надёжности графов средней размерности используя метод полного перебора совместно с CUDA.
- Размерность графа, для которого можно произвести расчёт за приёмлемое время, зависит от мощности и количества видеокарт, в большей степени от количества одновременно исполняемых потоков.

Варианты развития

- Оптимизация алгоритма полного перебора для экономии памяти.
- Предварительные упрощения структуры графа (для случайных графов).
- Численный эксперимент на MPI-кластере для сравнения с CUDA.
- Моделирование для более мощных кластеров.

Вопросы?