Т-СЕРВИСЫ
холдинг Т-Платформы

ПРОСТО ПОСТАВЬТЕ ЗАДАЧУ

Алгоритмические особенности создания многосеточного решателя СЛАУ на распределённых вычислительных системах с графическими ускорителями

Б.И. Краснопольский[1,2], *krasnopolsky@imec.msu.ru*

А.В. Медведев[2] *alexey.v.medvedev@gmail.com*

[1] НИИ механики МГУ, Москва
[2] ЗАО «Т-Сервисы», Москва

http://www.t-services.ru

Цели и задачи

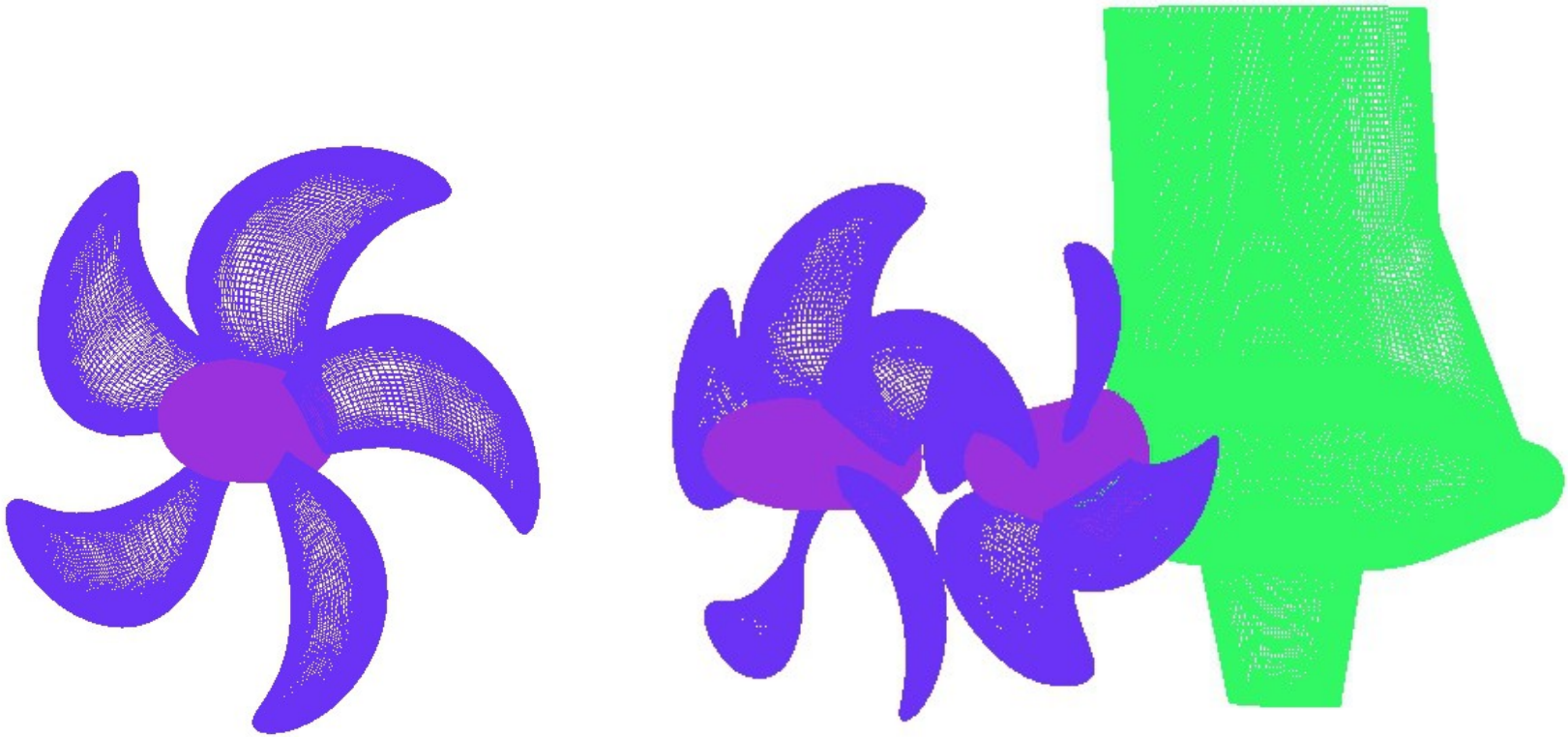
Эффективные методы решения эллиптических уравнений – необходимое условие для решения подавляющего большинства задач механики сплошных сред

Наиболее широко используемые методы для решения больших сильно-разреженных СЛАУ для эллиптических уравнений – итерационные методы подпространства Крылова и/или многосеточные методы

- Поиск и исследование алгоритмов распараллеливания вычислений в связи с реализацией на GPU

- Создание GPU-решателя, способного ускорить вычисления в ряде практических задач:
  - судостроения
  - многофазные многокомпонентные течения (нефтедобыча)

Моделирование напорных характеристик гребных винтов



OpenFOAM, расчётные сетки до 100 млн. ячеек

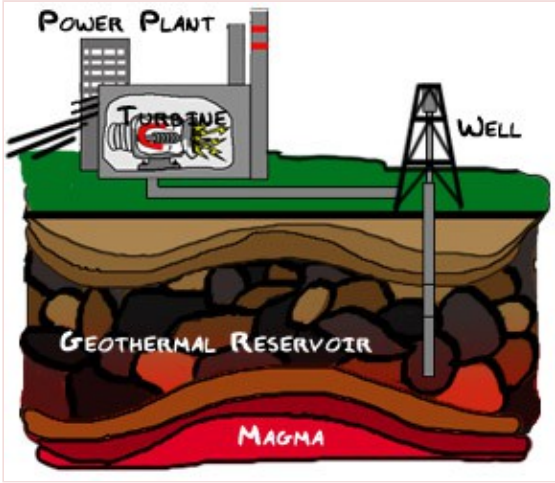
Моделирование многофазных многокомпонентных течений в пористой среде

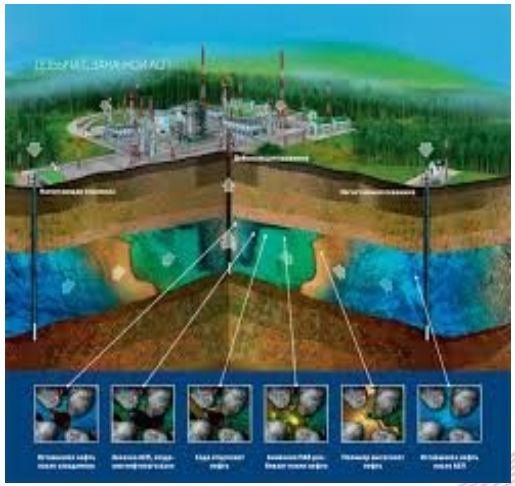
$$\frac{\partial}{\partial t}\left(m\sum_{i=1}^{3}\frac{M_{(1)}x_i}{v_i}s_i \right)+\operatorname{div}\left(\sum_{i=1}^{3}\frac{M_{(1)}x_i}{v_i}\mathbf{w}_i \right)=0$$

$$\frac{\partial}{\partial t}\left(m\sum_{i=1}^{3}\frac{M_{(2)}(1-x_i)}{v_i}s_i \right)+\operatorname{div}\left(\sum_{i=1}^{3}\frac{M_{(2)}(1-x_i)}{v_i}\mathbf{w}_i \right)=0$$

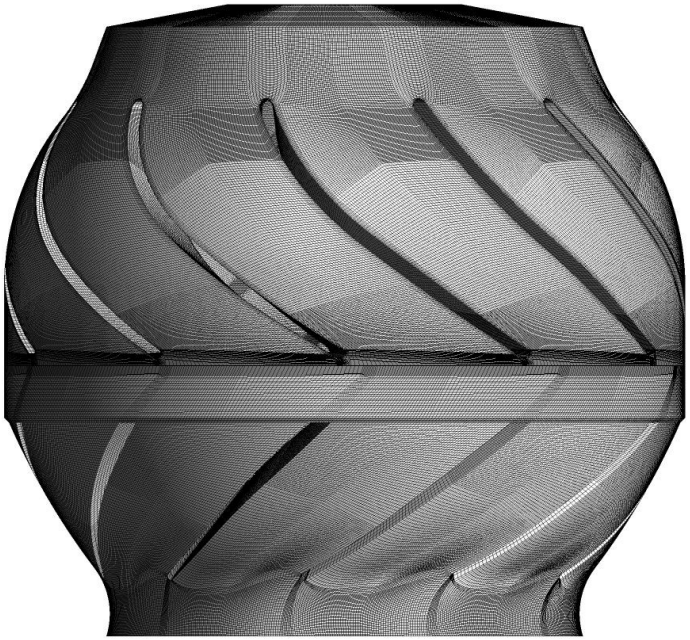
$$\frac{\partial}{\partial t}\left(m\sum_{i=1}^{3}\frac{e_i}{v_i}s_i+(1-m)\frac{e_s}{v_s} \right)+\operatorname{div}\left(\sum_{i=1}^{3}\frac{h_i}{v_i}\mathbf{w}_i-\lambda\frac{\partial T}{\partial \mathbf{r}} \right)=A_g$$

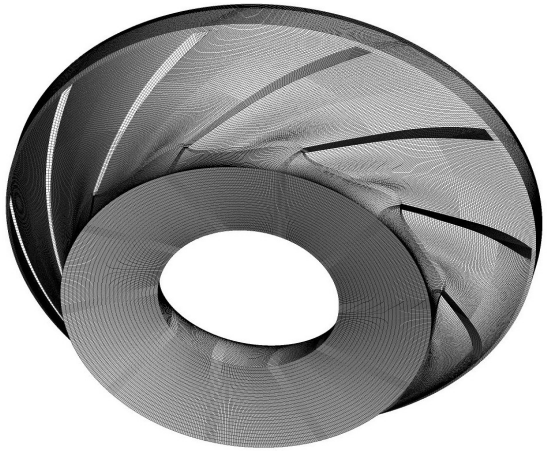
$$\mathbf{w}_i=-K\frac{f_i}{\mu_i}\left(\frac{\partial P_i}{\partial \mathbf{r}}-\frac{M_i}{v_i}\mathbf{g} \right)$$

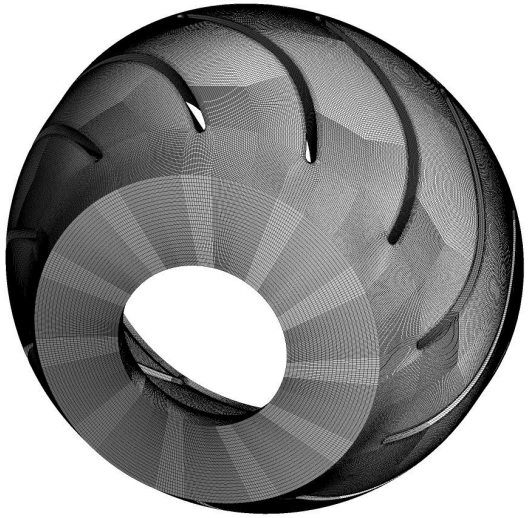




Тестовая задача: ступень погружного насоса







- Две детали: размер сеток 2.7 млн. и 4.9 млн. ячеек
- Сетка построена в ANSYS ICEM CFD, расчет проведен в пакете OpenFOAM

Итерационные методы

Solver: ***BiCGStab***

Preconditioner: ***Multigrid***, V-cycle

CAMG:
Classical Algebraic Multigrid

hypre-based implemenation

SAMG:
Smoothed Aggregation Multigrid

CUSP-based implementation

Smoother: ***Chebyshev polynomials***

Polynomial order: ***2***

Polynomial order: ***2, 3***

GPU как ускоритель

- Перенести код целиком невозможно
- Стратегия состоит в выделении ключевых блоков алгоритма и реализации в виде “ядер” CUDA
- Другая архитектура памяти – пересмотр всех алгоритмов
- Копирование данных в память GPU и обратно
  - вносит дополнительную латентность вызовов
  - касается и слоя MPI

Произведение разреженной матрицы на вектор (SpMV)

- Теоретический минимум операций чтения из памяти – **NNZ + N**
- Формат CRS: **2*NNZ + 2*N**
  - реализация Натана Белла (например, CUSP)
  - MGPU Sparse Library. URL: *http://www.moderngpu.com*
- Формат ELLPACK: **2S*N+N**, **min:2*NNZ+N**

  - реализация тривиальна

  - есть множество модификаций, в т.ч. “гибридные”
- Формат DIA: **ND*N + N**, **min: NNZ+N**

  - реализация тривиальна

  - нетривиален алгоритм преобразования в DIA из других форматов

  - *Возможно, “гибридные” модификации перспективны?*

Время выполнения SpMV: насос, 2.7M, GPU: M2090, 64-bit float, мсек. CUDA Toolkit 5.0

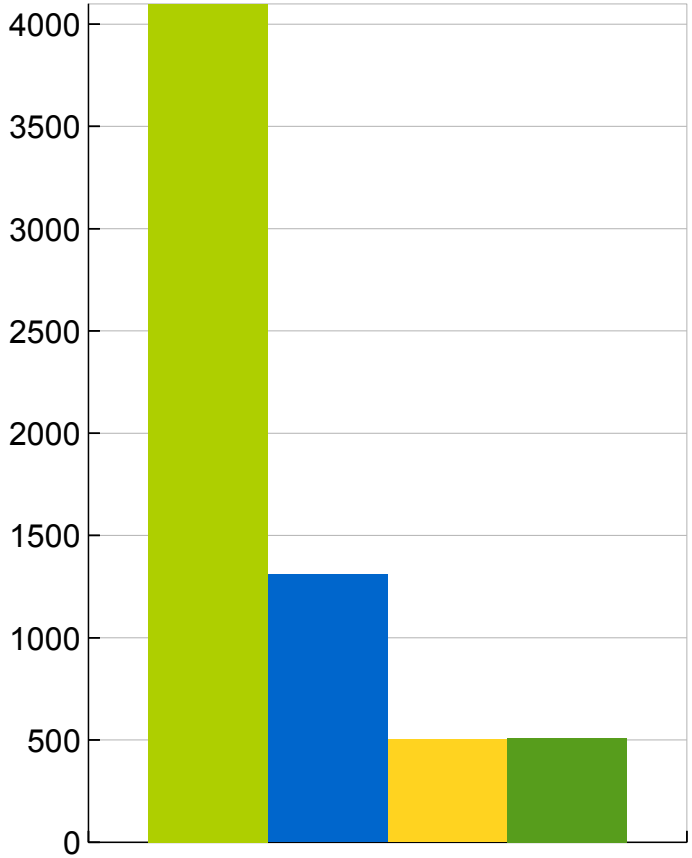
level:	csr (GPU, cusp)	ell (GPU, cusp)	hyb (GPU, cusp)	csr (GPU, CUSPARSE)	csr (1CPU)	CPU/best GPU ratio
0	3.25	2.23	2.23	3.55	38.04	17.06
1	3.92	4.63	3.94	3.53	35.10	9.93
2	2.74	3.32	2.37	2.23	20.44	9.15
3	1.18	1.91	1.39	1.49	12.74	10.77
4	0.51	1.57	0.94	0.93	6.95	13.72
5	0.21	0.78	0.53	0.41	3.08	14.60
6	0.08	0.31	0.44	0.21	1.11	14.64
7	0.03	0.19	0.47	0.07	0.31	10.79
8	0.01	0.11	0.26	0.04	0.09	8.76
9	0.004	0.05	0.06	0.03	0.02	4.38

Операции с векторами

- Используются функции CUSP и thrust:
  - cusp::blas::axpby()
  - thrust::reduce(), thrust::transform_reduce()
- Векторы на GPU представляются как объекты-отражения векторов на CPU
- Прозрачное копирование объектов с CPU на GPU и обратно позволяет сопрягать CPU и GPU код в гибких комбинациях

BiCGStab CUSP/SLS на 1 GPU

Время выполения, мсек:



Задача: направляющий аппарат 4.9M

CPU: **Intel Xeon E5-2665**

GPU: **NVIDIA Tesla M2090**

20 итераций

1 CPU socket / 1 GPU = 2.6

BiCGStab CUSP на 1 GPU: теоретический максимум

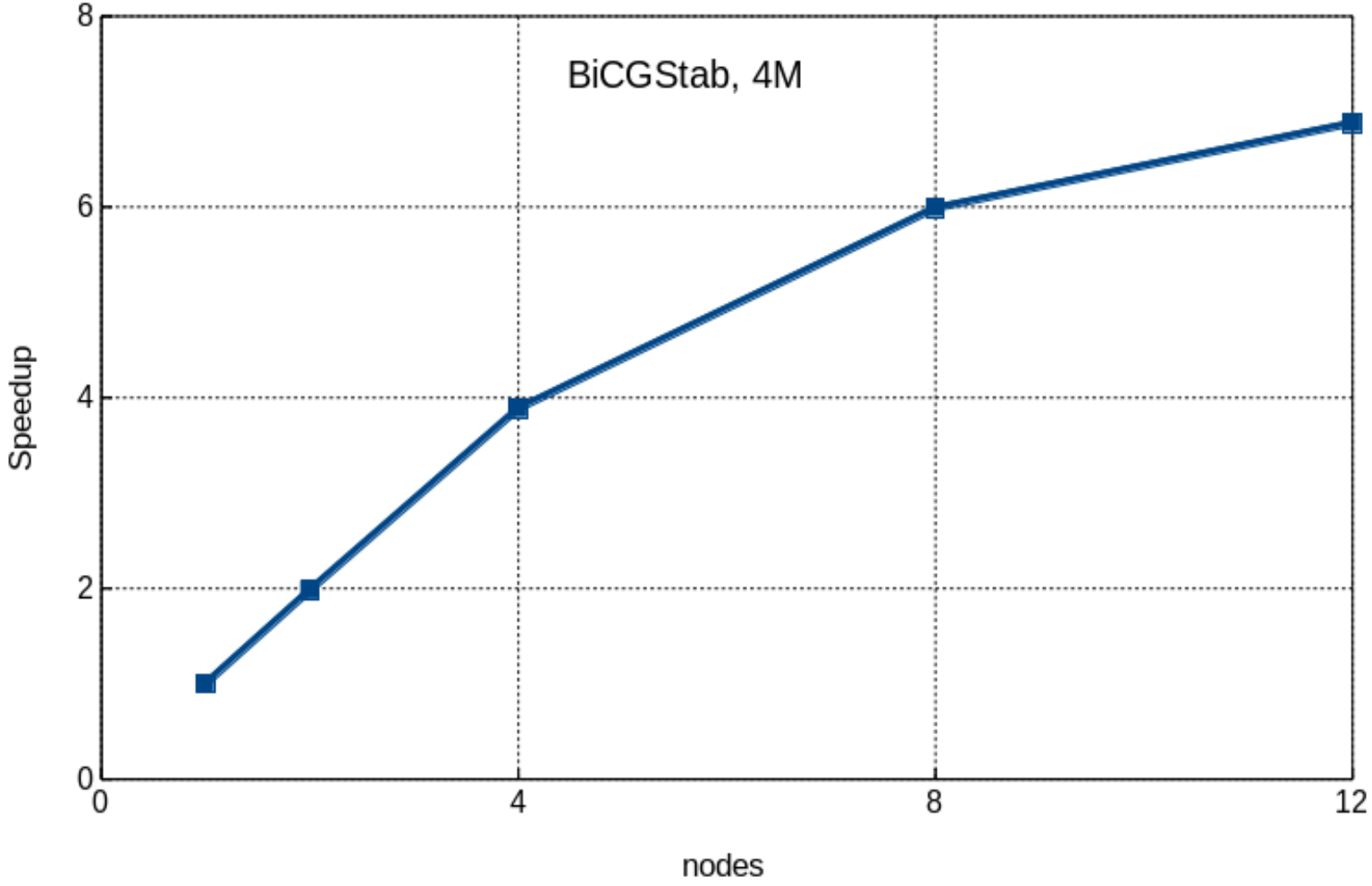
- Задача относится к категории memory-bound

  Max. bandwith Intel Xeon E5-2665, 1 socket: 51,2 Gb/s
  Max. bandwith NVIDIA Tesla M2090: ECC off: 177 Gb/s
  ECC on: 154,9 Gb/s

- Соотношение: **3,0** – это теоретический максимум

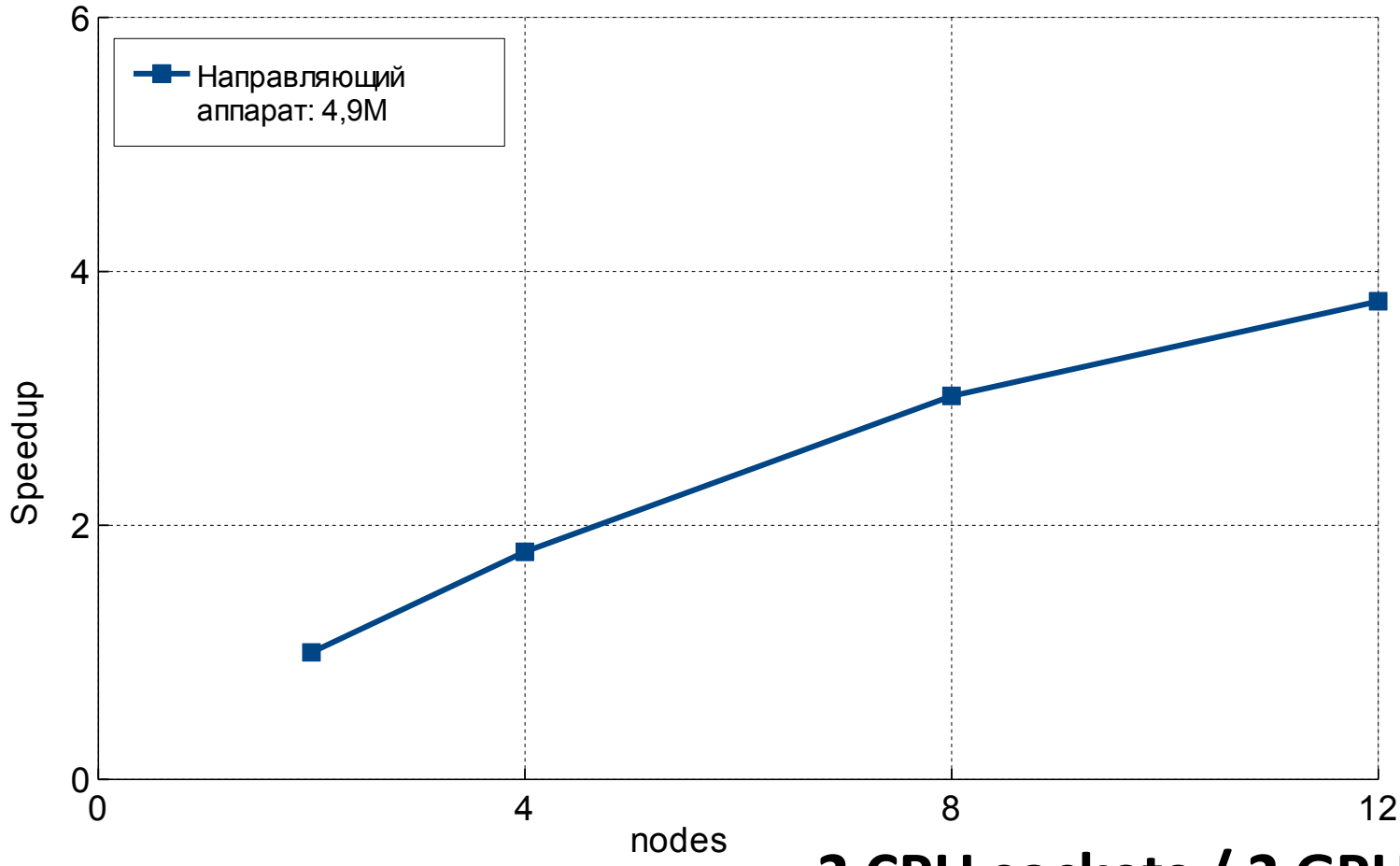
Масштабируемость BiCGStab SLS_GPU

Задача: напрвляющий аппарат 4.9M



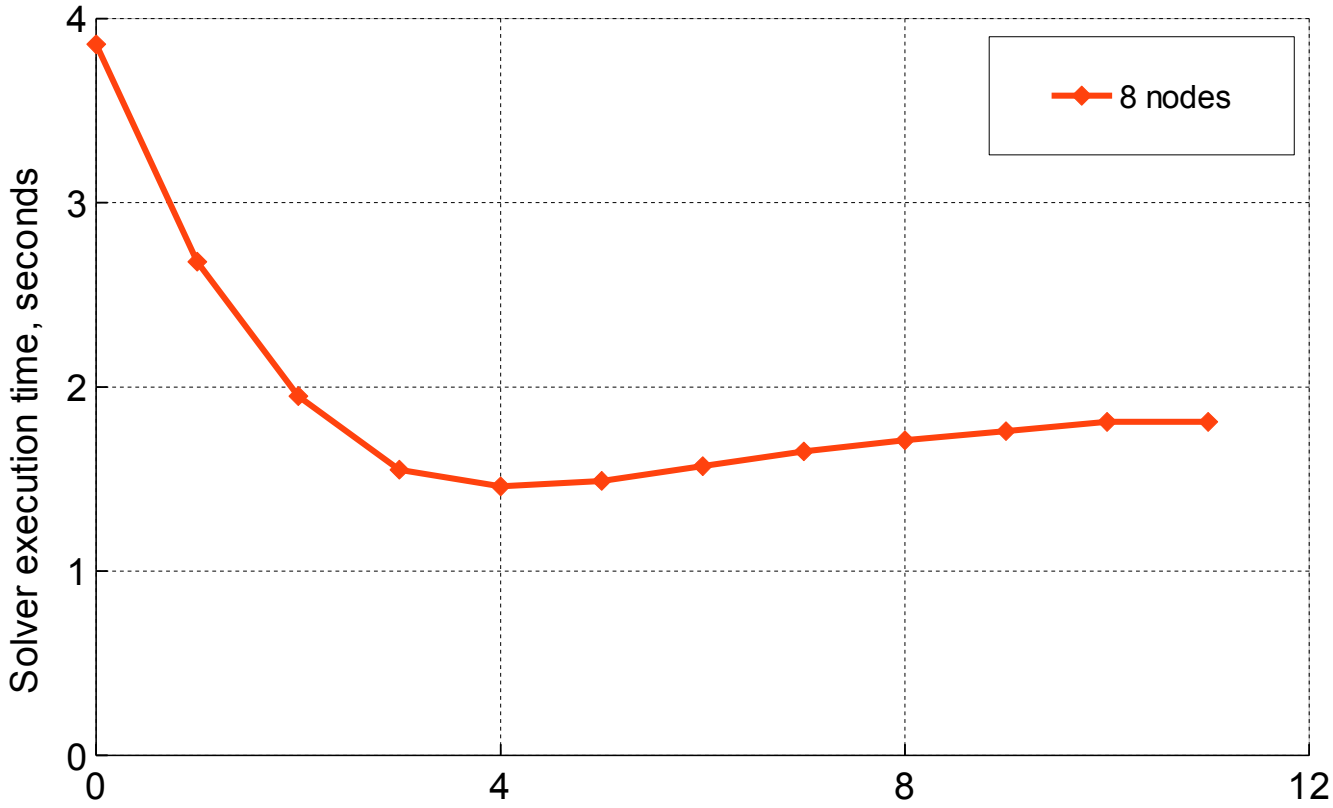
CAMG солверная часть, масштабируемость

Задача: направляющий аппарат 4.9M



2 CPU sockets / 2 GPUs = 1.7

CAMG: несколько уровней MG на GPU, 8 nodes



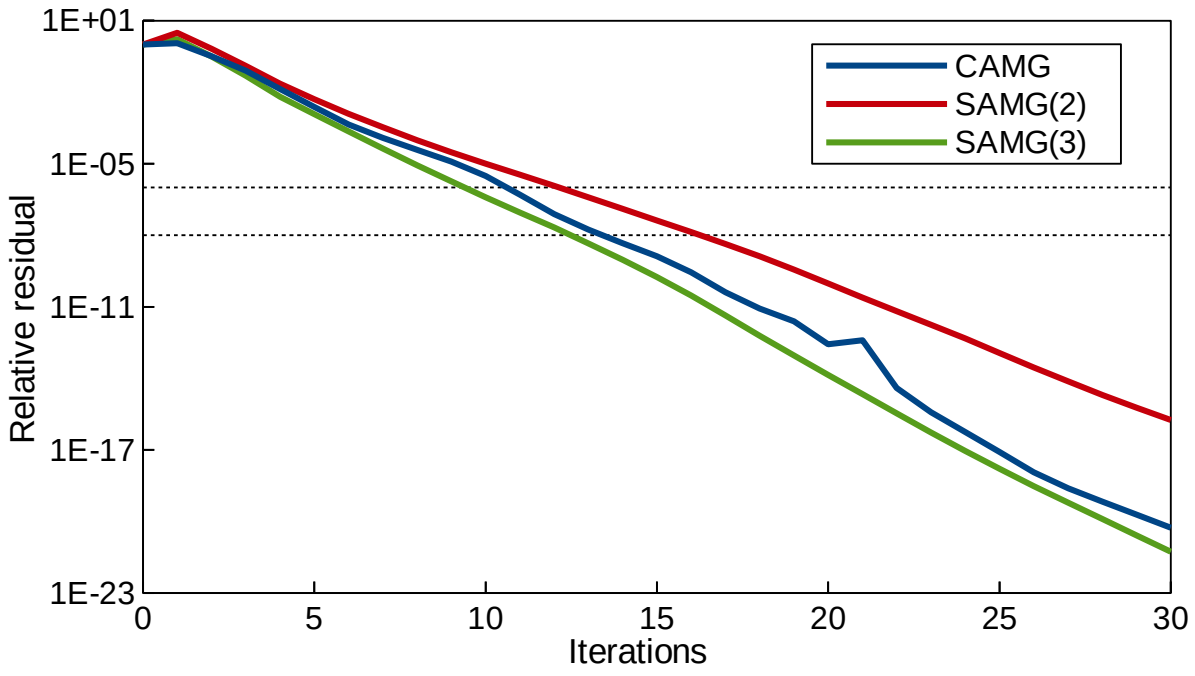
- Оптимально: 4 уровня MG на GPU, остальные на CPU
- Другое соотношение для другого числа узлов и размера матрицы

Скорость сходимости методов, рабочее колесо 2.7M

#	Размер матрицы	
	CAMG	SAMG
0	2712320	2712320
1	1078627	364846
2	427949	20661
3	181283	1132
4	75393	107
5	28974	24
6	10320	-
7	3464	-
8	1123	-
9	356	-
10	109	-
11	46	-

SAMG:

+ вдвое быстрее скорость агрегирования при переходе на следующий уровень
+ быстрее построение иерархии матриц
- медленнее скорость сходимости



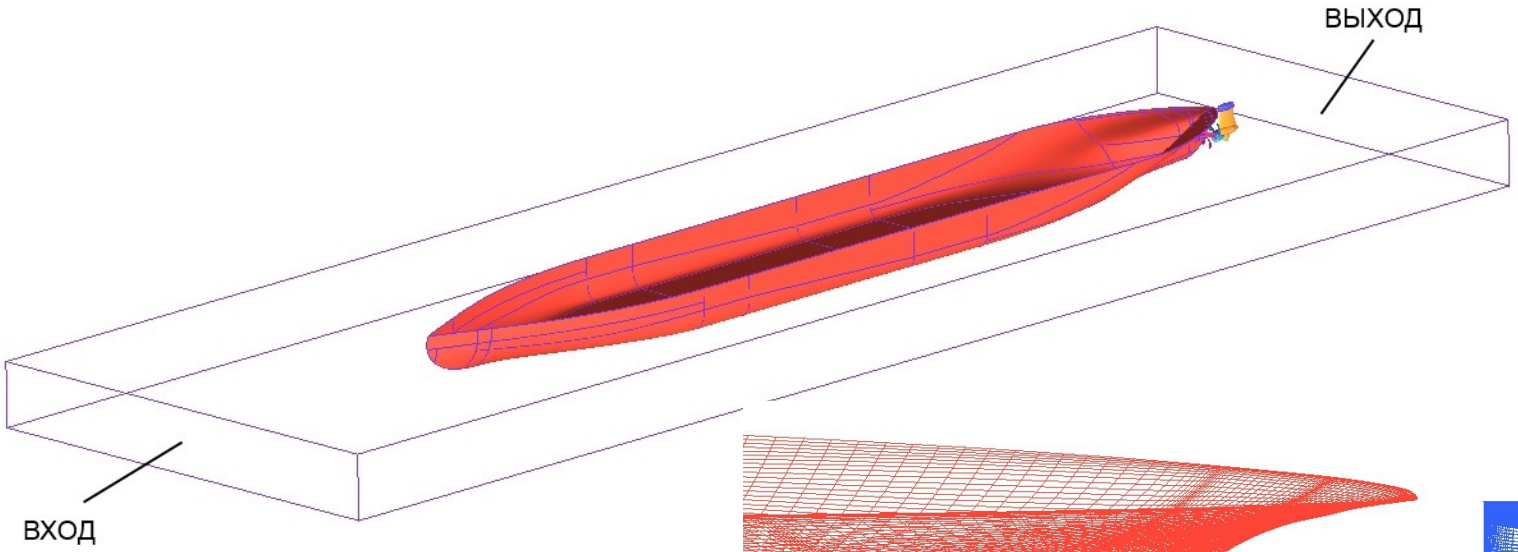
Результаты, невязка 10^{-8}

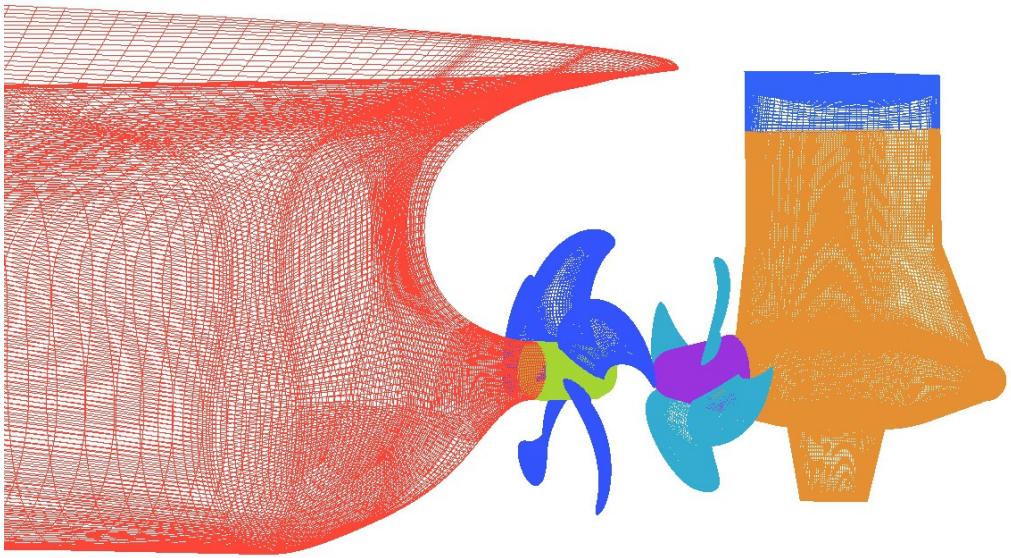
Матрица	Рабочее колесо, 2.7 млн. неизвестных			Направляющий аппарат, 4.9 млн. неизвестных		
Метод	CAMG	SAMG(2)	SAMG(3)	CAMG	SAMG(2)	SAMG(3)
Время расчета 1 итерации	3.45	2.23	2.88	6.16	4.02	5.23
Кол-во итераций	14	17	13	16	19	17
Время решения	48.3	37.9	37.4	98.6	76.4	88.9

Сопряжение с OpenFOAM

- Создан плагин, позволяющий использовать реализованные методы в ходе расчетов задач в пакете OpenFOAM, в т.ч. MiltiGPU-режим
- Реализована поддержка ***processor*** и ***cyclicAMI***-патчей для расчета задач с вращающимися подобластями в параллельном режиме
- Используемые методы обладают большей робастностью по сравнению с многосеточным методом GAMG из пакета OpenFOAM

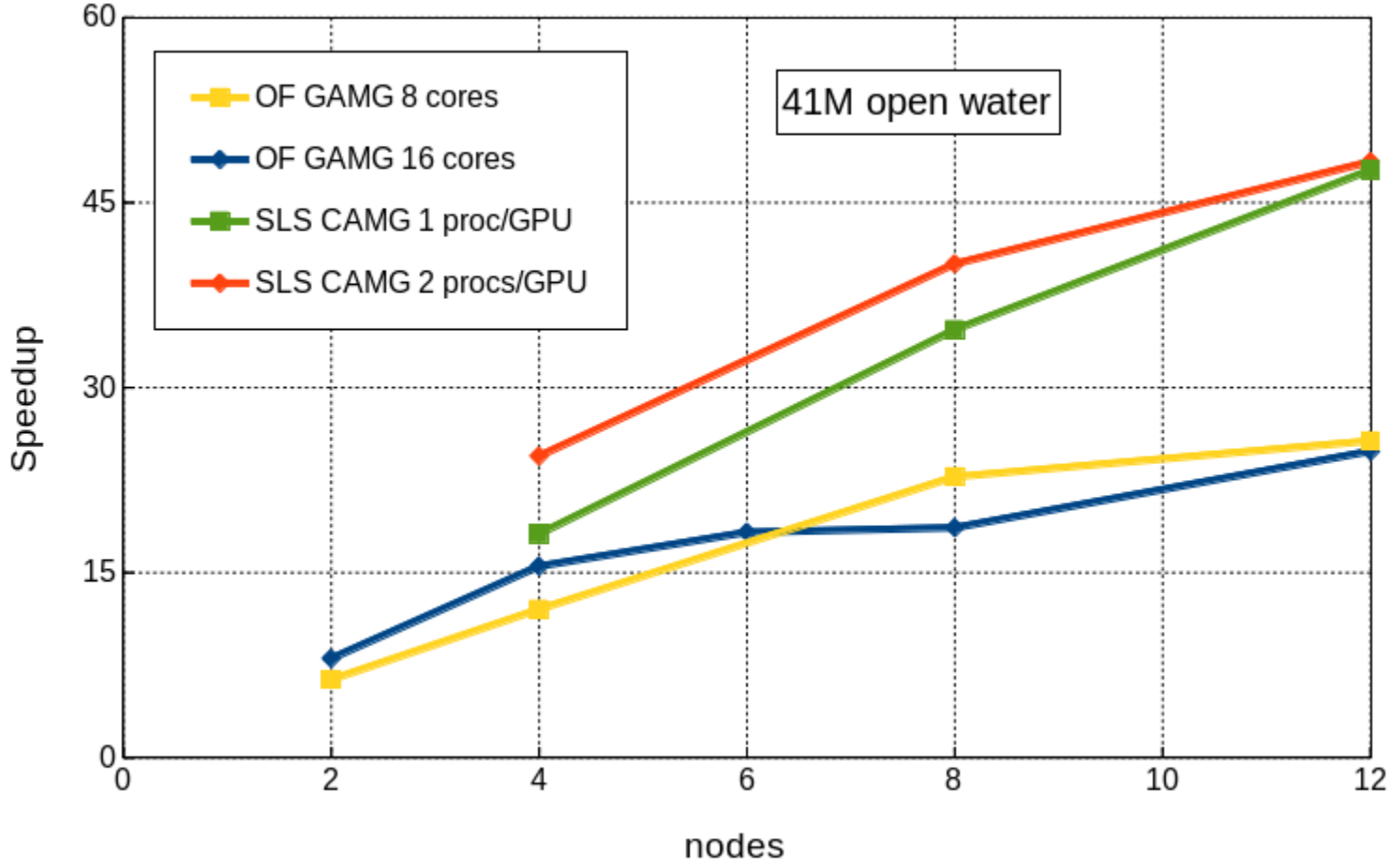
Моделирование напорных характеристик винтов





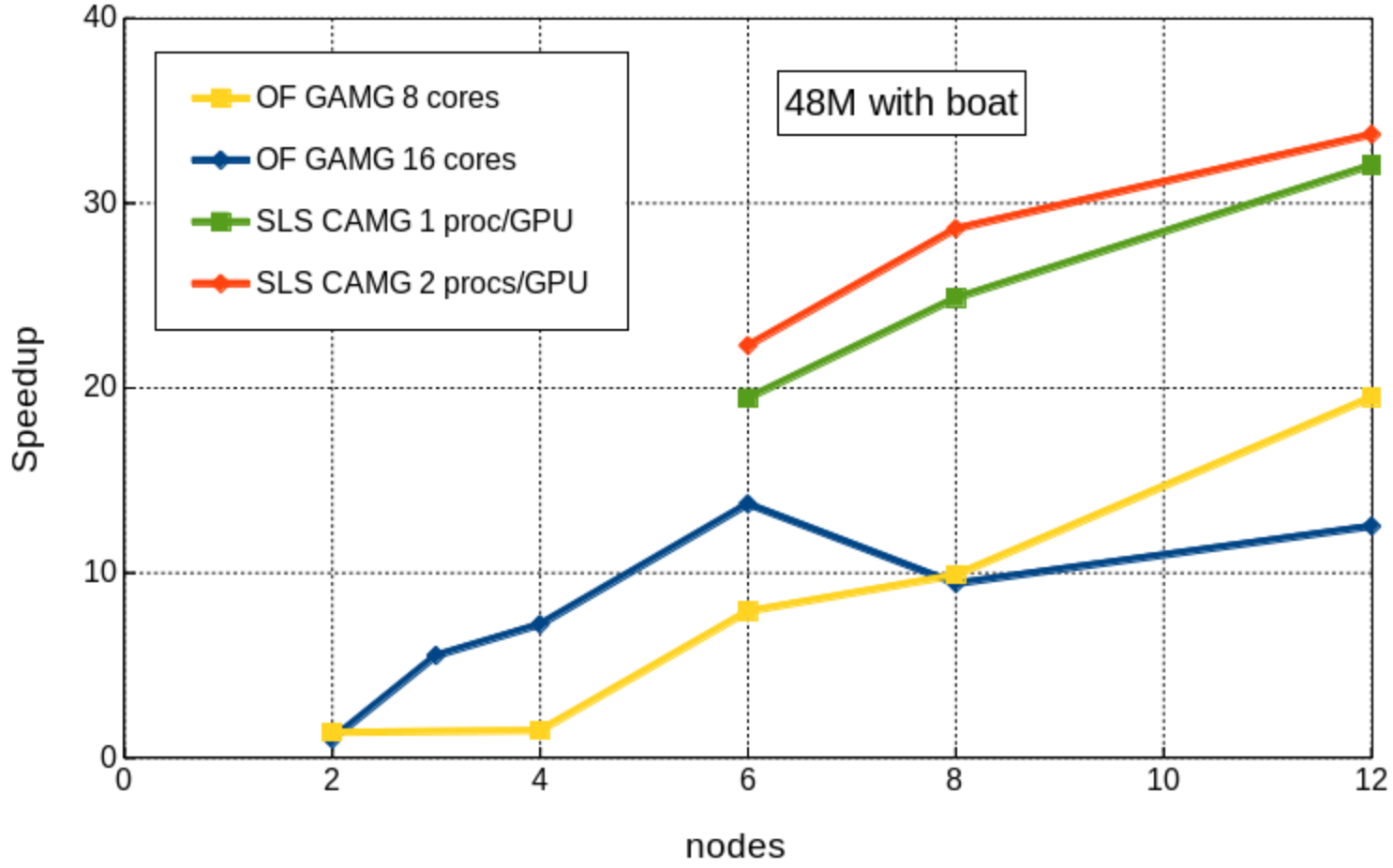
Моделирование характеристик винтов в OF: постановка задачи в открытой воде

CPU: Intel Xeon E5-2665; GPU: NVIDIA Tesla M2090



Моделирование характеристик винтов в OF: постановка задачи с корпусом судна

CPU: Intel Xeon E5-2665; GPU: NVIDIA Tesla M2090

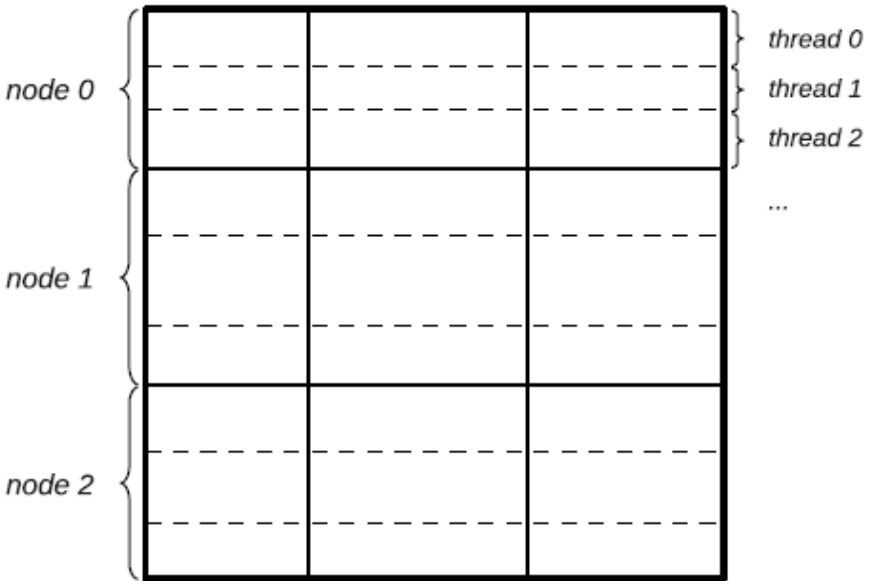


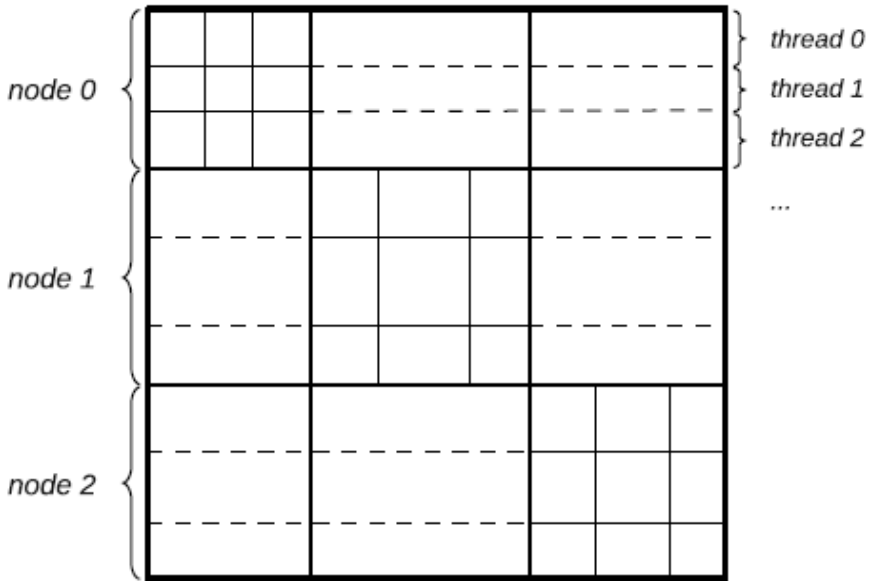
Перспективы дальнейшего развития

- Поиск форматов представления матриц в применении к GPU, разработка гибридных форматов
- Уменьшение объёмов чтения из основной памяти: 16-битные индекс-векторы, 32-битные числа с плавающей точкой
- Перенос setup-части
- Совершенствование схемы сопряжения с OpenFOAM
- Адаптация к Kepler
- ...

Спасибо за внимание!

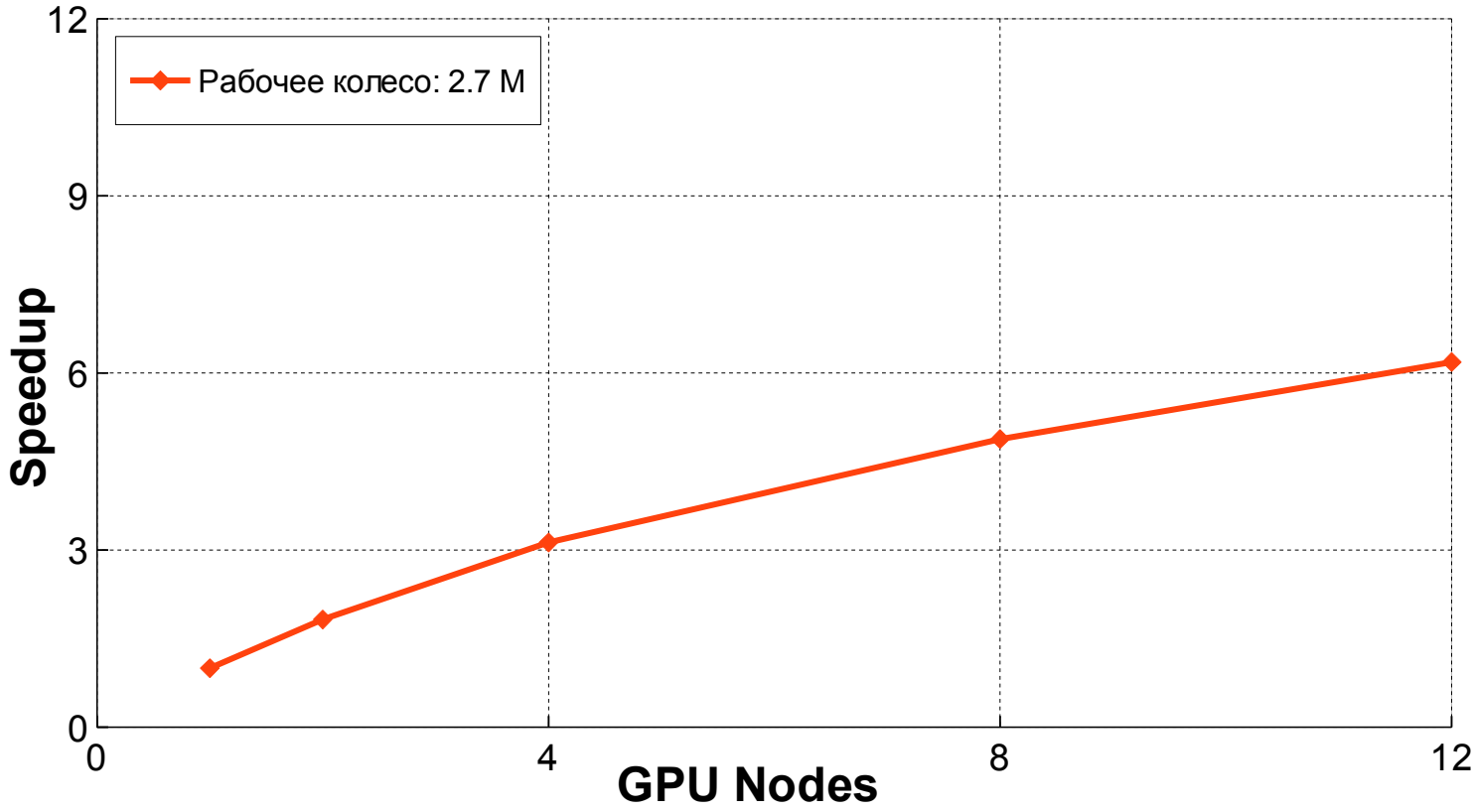
SparseLinearSol (SLS), иерархическая структура





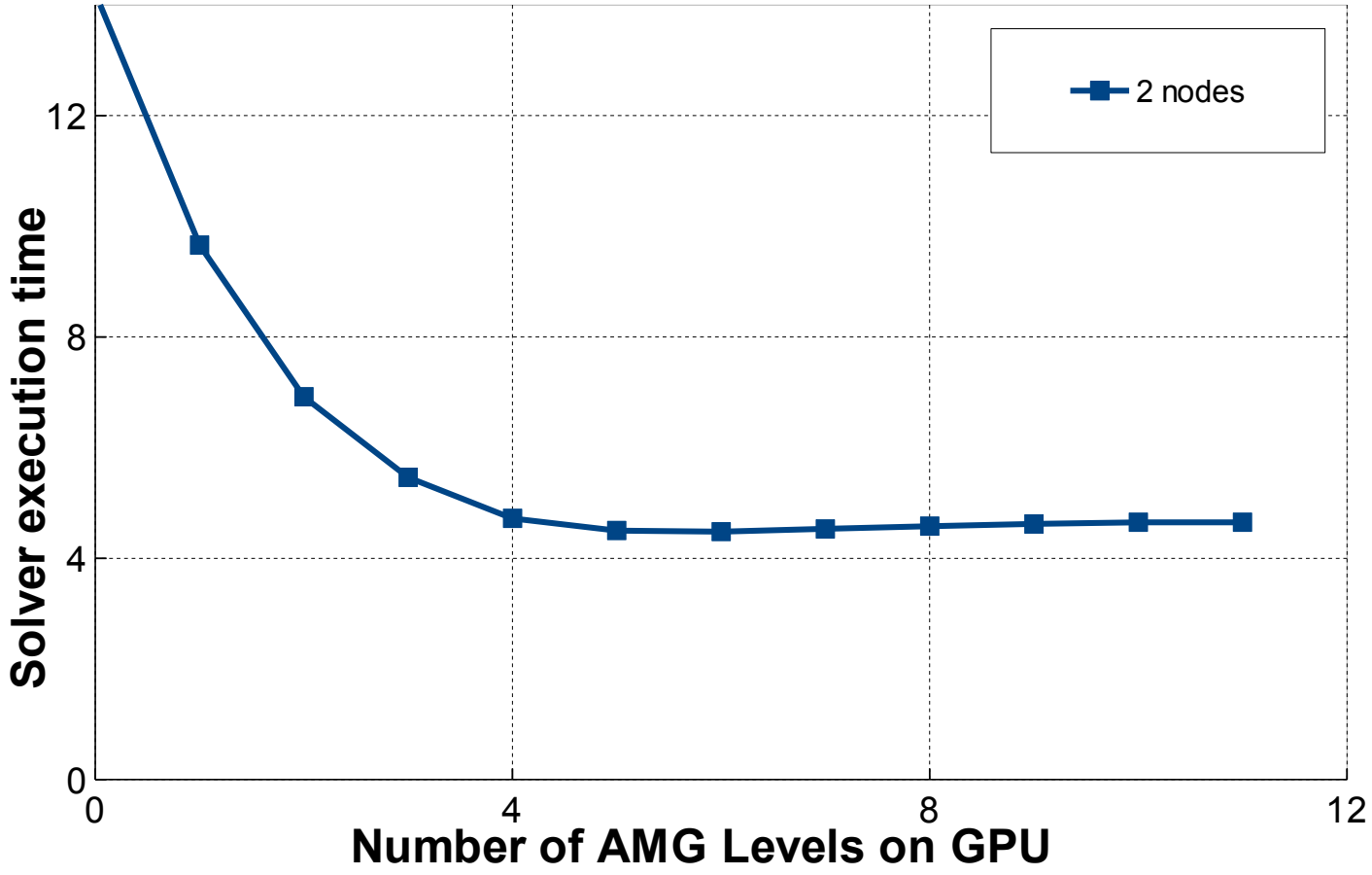
- Многоуровневое разделение данных внутри узла
- На каждом уровне разделения своя реализация синхронизации процессов

CAMG solver part 2M scalability



1 CPU socket / 1 GPU = 1.8

CAMG: несколько уровней на GPU, 2 nodes



Выводы, дальнейшая работа

- Большое количество алгоритмических сложностей:
  - поиск форматов в применении к GPU, разработка гибридных форматов
  - уменьшение объёмов чтения из основной памяти: 16-битные индекс-векторы, 32-битные числа с плавающей точкой
  - GPU-алгоритмы сегментирования
- Ближайшие планы: multi-GPU AMG солвер
- Интеграция с OpenFOAM
- Адаптация к Kepler...