Статико-динамическое распараллеливание для новых аппаратных платформ

Новые методы и приложения

В.А.Роганов, НИИ Механики МГУ

3 марта 2015 г.

Проблемные области и новый подход

HPC-направление сегодня является чрезвычайно важным для практики. Различные наукоемкие приложения, суперкомпьютерное моделирование, BigData-аналитика, обработка текстовой информации, обработка видео - все эти области все активнее начинают задействовать технологии HPC.

Объединив методы динамического и статического распараллеливания программ удалось получить новое качество, которого во многих случаях достаточно для решения наиболее сложных на сегодняшний день проблем: эффективного задействования большого количества счетных ядер и векторизации вычислений.

Содержание доклада

1. Статическое и динамическое распараллеливание.
2. Новые и перспективные аппаратные HPC-платформы.
3. Гибридное статико-динамическое распараллеливание. Проактивные вычисления и сверхмемоизация.
4. Как решаются важные для практики задачи.
5. Что уже есть по теме и возможные точки роста.

Статическое распараллеливание

Статическое распараллеливание обеспечивает максимальную эффективность в условиях, когда конфигурация ресурсов и входных данных задачи полностью определены до начала счета.

Большинство параллельных алгоритмов базируются именно на этой парадигме, которая, в том числе, позволяет эффективно использовать вычислительную мощность векторных вычислительных устройств.

Большинство MPI-программ реализуют параллельные алгоритмы, где вся динамика сводится к адаптации к числу доступных счетных ядер. Также старые GPGPU-устройства поддерживают только статический параллелизм.

Динамическое распараллеливание

Динамическое распараллеливание было и остается эффективным средством, позволяющим задействовать большое количество вычислительных ядер.

Оно позволяет обеспечить гибкую балансировку нагрузки в условиях, когда либо объемы вычислений, либо доступные ресурсы вычислительной платформы неоднородны и непостоянны во время счета.

Динамическое распараллеливание

Языки программирования и их расширения

- SISAL
- Charm++
- Cilk
- CUDA и OpenCL (новые версии)
- T-система

Вариации/библиотеки на тему

- ThreadPool
- SuperCall

Пример: как пишут на T++

Материал из Википедии — свободной энциклопедии

T++ — язык программирования … расширяющий язык C++ несколькими словами, указывающими на возможность проведения параллельных вычислений:

```
tfun int fib(int n) {
  return n < 2 ? n : fib(n-1) + fib(n-2);
}
tfun int main (int argc, char *argv[]) {
  if (argc!=2) {printf("Usage: fib <n>\n"); return 1;}
  int n = atoi(argv[1]);
  printf("fib(%d) = %d\n", n, (int)fib(n));
  return 0;
}
```

Истоки Т-системы: “звездные войны”

Современная интерпретация :)

Новые аппаратные платформы: ускорители

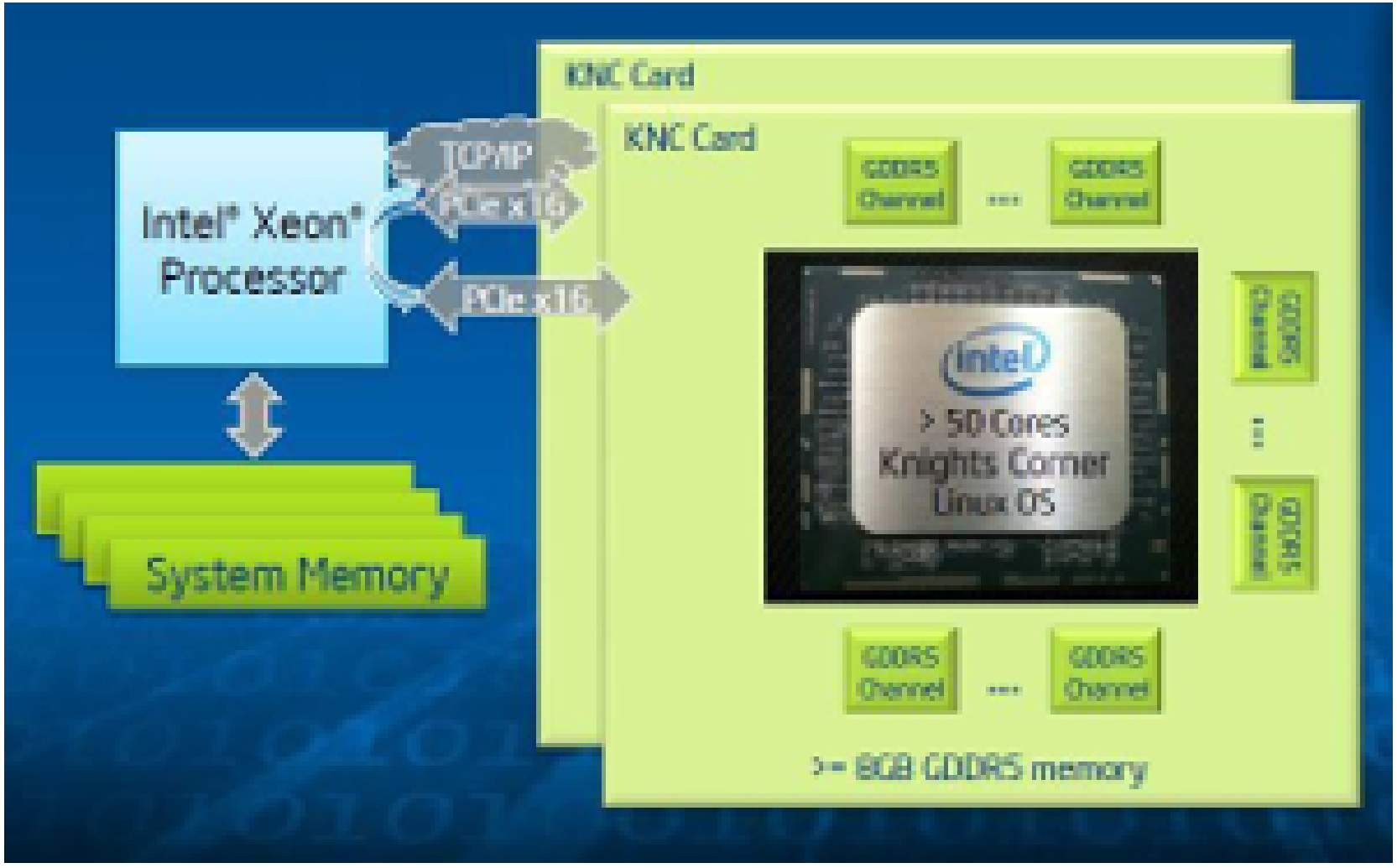
Сопроцессор Intel Xeon Phi

Сопроцессоры Intel Xeon Phi, созданные на базе архитектуры Intel MIC, работают на операционной системе Linux, поддерживают модель памяти x86 и арифметику с плавающей точкой IEEE754.

Они могут непосредственно исполнять приложения, написанные на стандартных языках программирования, таких как C, C++, FORTRAN.

Приложения для сопроцессора могут быть разработаны при помощи интегрированной среды разработки Intel Composer XE 2013, которая включает в себя компиляторы и библиотеки, математические библиотеки высокой производительности, различные инструменты и отладчики.

Аппаратное устройство



Аппаратные возможности

Несколько сопроцессоров Intel Xeon Phi могут быть установлены в одной компьютерной системе (на одном host-узле).

В рамках единой системы сопроцессоры могут обмениваться информацией друг с другом через PCIe-соединение (взаимодействие «точка–точка») без какого-либо вмешательства со стороны host-узла.

Аналогичным образом сопроцессоры могут обмениваться данными через сетевую карту, такую как InfiniBand или Ethernet.

CUDA и OpenCL

Сравнение пиковой производительности

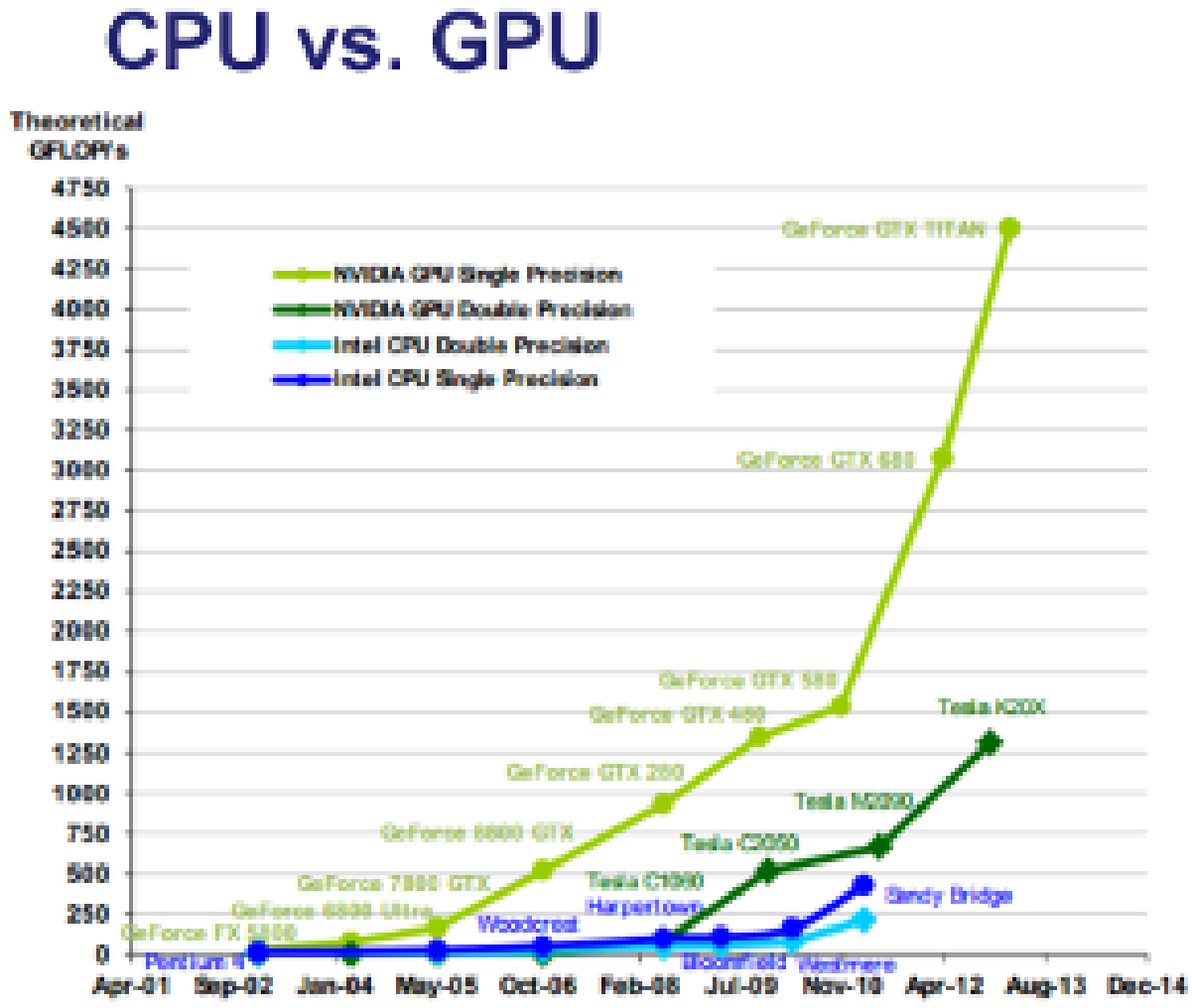


Figure 1 Floating-Point Operations per Second for the CPU and GPU

Новые возможности CUDA 6.*, OpenCL 2.*

CUDA 6.0 – 6.5 поддерживает:

- Общее адресное отображения для CPU и GPU
- Динамический параллелизм (запуск новых ядер из уже работающих ядер)
- Возможность использования GPU из нескольких параллельных потоков CPU

OpenCL 2.0:

- Shared Virtual Memory
- Nested parallelism (выполнение kernel без привлечения host)
- pipe / atomic операции

Бонус: OpenCL может компилироваться для FPGA

Современные видеоадаптеры

	GEFORCE GTX 980
CUDA Cores	2048
Base Clock	1126 MHz
Boost Clock	1216 MHz
Single Precision	5 Teraflops
Memory Config	4GB / 256-bit GDDR5
Memory Speed	7.0 Gbps
Power Connectors	6-pin + 6-pin
TDP	165W

Как быть с реализацией динамического параллелизма на ускорителях

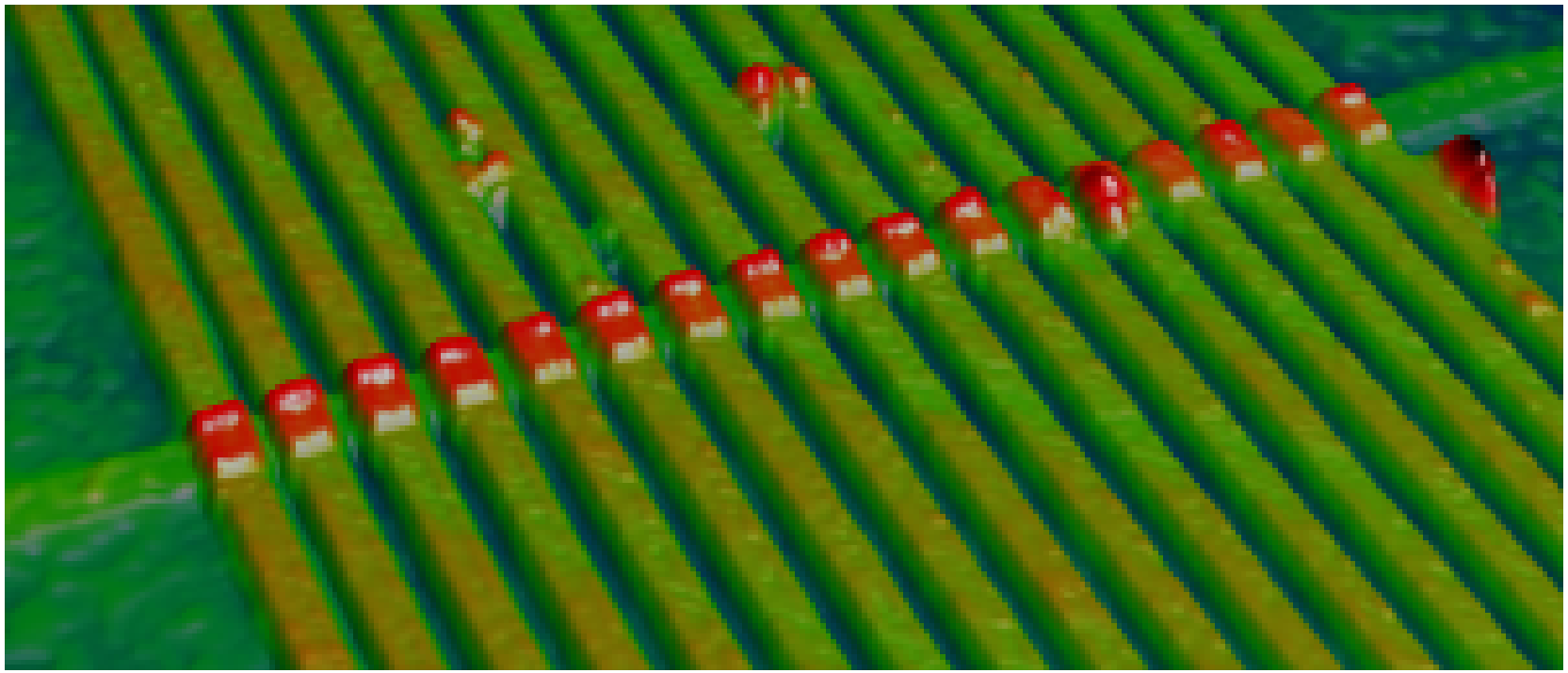
На аппаратных ускорителях далеко не всегда есть возможности явного сохранения и восстановления контекста, типа makecontext, swapcontext

Вопрос: нужен ли вообще DP на акселераторах?

Ответ: желателен - есть задачи, в которых гранулы параллелизма столь легкие, что накладные расходы на взаимодействие CPU<->GPU портят жизнь.

В конечном счете наиболее эффективно статико-динамическое распараллеливание программ.

Мемристоры + оптические компоненты для перспективных моделей суперЭВМ



Гибридное статико-динамическое распараллеливание программ

“Статико-динамическое”

- Если упорядочивать слова в названии по тяжести счета

“Динамико-статическое”

- Если упорядочивать по уровню подзадач

Разные платформы требуют разные подходы!

Сверхмемоизация и проактивность

Автоматическое динамическое квантование в пространстве подзадач. Применима в т.ч. для императивного кода.

1. Разложение подзадачи в композицию

T(n) = V + R [+ T(n+1)]

2. Перестройка (трансфигурация)

n -> n+1

- Для GRID/Cloud-систем: кооперативное планирование V-частей
- Для CUDA: агрегирование + многопоточная обработка остаточных подпрограмм (R-частей). Доступно 3% вычислительной мощности.

Танец маленьких подпрограмм

Комбинация сверхмемоизации с динамическим распараллеливанием и позволяет получить эффект автоматической динамической векторизации вычислений. Статическое можно рассматривать как ”замерзшее” динамическое.

Автоматическая векторизация вычислений

Вычисление fib(10):

.10 *10 .9 .8 *8 .7 .6 *6 .5 .4 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2
.1 .0 *5 .4 .3 *3 .2 .1 *2 .1 .0 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2
.1 .0 *7 .6 .5 *5 .4 .3 *3 .2 .1 *2 .1 .0 *4 .3 .2 *2 .1 .0 *3
.2 .1 *2 .1 .0 *0 *1 *1 *0 *1 *0 *1 *1 *0 *1 *1 *0 *1 *0 *1
*1 *0 *1 *1 *0 *1 *6 .5 .4 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2 .1 .0
*5 .4 .3 *3 .2 .1 *2 .1 .0 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2 .1 .0
*9 .8 .7 *7 .6 .5 *5 .4 .3 *3 .2 .1 *2 .1 .0 *4 .3 .2 *2 .1 .0
*3 .2 .1 *2 .1 .0 *0 *1 *1 *0 *1 *0 *1 *1 *0 *1 *1 *0 *1 *0
*1 *1 *0 *1 *1 *0 *1 *6 .5 .4 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2
.1 .0 *5 .4 .3 *3 .2 .1 *2 .1 .0 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2
.1 .0 *8 .7 .6 *6 .5 .4 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2 .1 .0 *5
.4 .3 *3 .2 .1 *2 .1 .0 *0 *1 *1 *0 *1 *1 *0 *1 *0 *1 *1 *0
*1 *0 *1 *1 *0 *1 *1 *0 *1 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2 .1
.0 *7 .6 .5 *5 .4 .3 *3 .2 .1 *2 .1 .0 *4 .3 .2 *2 .1 .0 *3 .2
.1 *2 .1 .0 *6 .5 .4 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2 .1 .0 *5 .4
.3 *3 .2 .1 *2 .1 .0 *0 *1 *1 *0 *1 *1 *0 *1 *0 *1 *1 *0 *1
*0 *1 *1 *0 *1 *1 *0 *1 *4 .3 .2 *2 .1 .0 *3 .2 .1 *2 .1 .0
*0 *1 *1 *0 *1 =55

Задачи, естественно возникшие из практики и некоторые полученные результаты

1. Криптоаналитические приложения
2. Теплогидродинамика
3. Решения СЛАУ с разреженной матрицей
4. Генетические алгоритмы (синтез/оптимизация)
5. Стабилизация и оптимальное управление
6. Сверхбыстрый анализ потоков данных

Публикация в CUDA-Альманахе

“Методы адаптации системы параллельного программирования OpenTS для поддержки работы T-приложений на гибридных вычислительных кластерах” в журнале Программные системы.

NVidia неожиданно включила нашу статью в альманах

Дешифратор паролей

Все помнят про опубликованные списки паролей? Конкатенация пароля с солью - защита от радужных таблиц - неэффективна при наличии у злоумышленника мощной ЭВМ.

Движки сайтов (Joomla и т.д.) имеют уязвимости в своих расширениях, поддерживаемых случайными разработчиками. Утечка зашифрованных паролей через SQL-инъекцию приводит к серьезным проблемам.

Хорошие словари паролей

Множество реально используемых паролей имеет неслучайное распределение. Чем лучше мы будем учитывать “традиции” составления паролей (то есть свод правил для составления цепочек символов), тем эффективнее можно организовать перебор вариантов.

Общая схема алгоритма приложения md5t

Использование и синтез оценочных функций для паролей

- Вводится понятие цены цепочки символов
- Обучение “традиции” на основе заданного текста

Статико-динамическое распараллеливание

- Динамическое распараллеливание на кластерном уровне
- Предвычисленная таблица списков суффиксов
- Динамическое распределение счетных гранул по ядрам GPU

Код реализации MD5

```
void md5(uint8_t* msg, unsigned len, uint32_t h[4]) {
    uint32_t blk[16];
    memset(blk,0,64); memcpy(blk,msg,len);
    blk[14] = len<<3; ((uint8_t*)blk)[len] = 0x80;
    uint32_t a = h[0]=0x67452301, b = h[1]=0xefcdab89,
             c = h[2]=0x98badcfe, d = h[3]=0x10325476;
#define RT(a,b,expr,k,s,t) \
    a += (expr)+(t)+blk[k]; a = b+(a<<s|a>>(32-s));
#define R(a,b,c,d,k,s,t)  RT(a,b,d^(b&(c^d)),k,s,t)
    R(a, b, c, d,  0,  7, 0xd76aa478)
    R(d, a, b, c,  1, 12, 0xe8c7b756)
    R(c, d, a, b,  2, 17, 0x242070db)
    R(b, c, d, a,  3, 22, 0xc1bdceee)
    R(a, b, c, d,  4,  7, 0xf57c0faf)
    /////   Еще 70 строчек в том же духе  //////
#undef R
#undef RT
    h[0]+=a; h[1]+=b; h[2]+=c; h[3]+=d;
}
```

Полный код вычислительного CUDA-ядра

```
GBL void md5kernelDyn(GPU_Job::jb_t& b, HCmp* hcmp, bool
  const size_t id = blockIdx.x*blockDim.x + threadIdx.x;
  char s[11] = {0,};
  unsigned cnt = b.p[1]; //aligned cnt
  assert(!(cnt % THBLK));
  unsigned todo = cnt/THBLK;
  unsigned *p = b.p + 2; //skip r&a cnts
  for(int k=0;k<6;k++) s[k]=b.s12[k];
  for (int i=0; i<todo; i++) {
    uchar* sfx=(uchar*)(p+id*todo+i);
    for(int k=0;k<4;k++) s[k+6] = sfx[k];
    unsigned len=0;
    for (int k=0; k<10; k++) len+=(((char*)s)[len]!=0);
    uint32_t h[4] = { -1u, };
    md5((uint8_t*)s,len,h);
    int pi = (*hcmp)(h); //password index
    if (!(pi<0))
      { printf("PASSWORD[%d]: %s\n",pi,s); found = true;
  }
}
```

Пример запуска (1 видеокарта ценой $100)

```
Reading hash info
[0]: 69996ae2e822cc1f18404d4c66cc4932
.....................................
[11]: ea332c272f7e6498fbbdfe538feb9a1d
Got 12 hashes
PASSWORD[0]: adamovich
PASSWORD[1]: gmatveev
PASSWORD[2]: baranessa
PASSWORD[3]: biliberda
PASSWORD[9]: inikogda
PASSWORD[11]: moyparol
PASSWORD[8]: nizachto
PASSWORD[5]: parolchik
PASSWORD[7]: vslomati
```

Время счета: 65 минут

Преимущества и перспективы md5t

- Компактный, расширяемый код (< 500 строк)
- Произвольное комбинирование хэш-функций со строчными операциями
- Эффективная работа на гибридных кластерах (двухуровневое динамическое распараллеливание для CPU+GPU)
- Динамическая адаптация к языковым “традициям”
- Динамическое изменение функции-оценки и глубины счета в зависимости от успешности уже декодированных паролей

Некоторые выводы

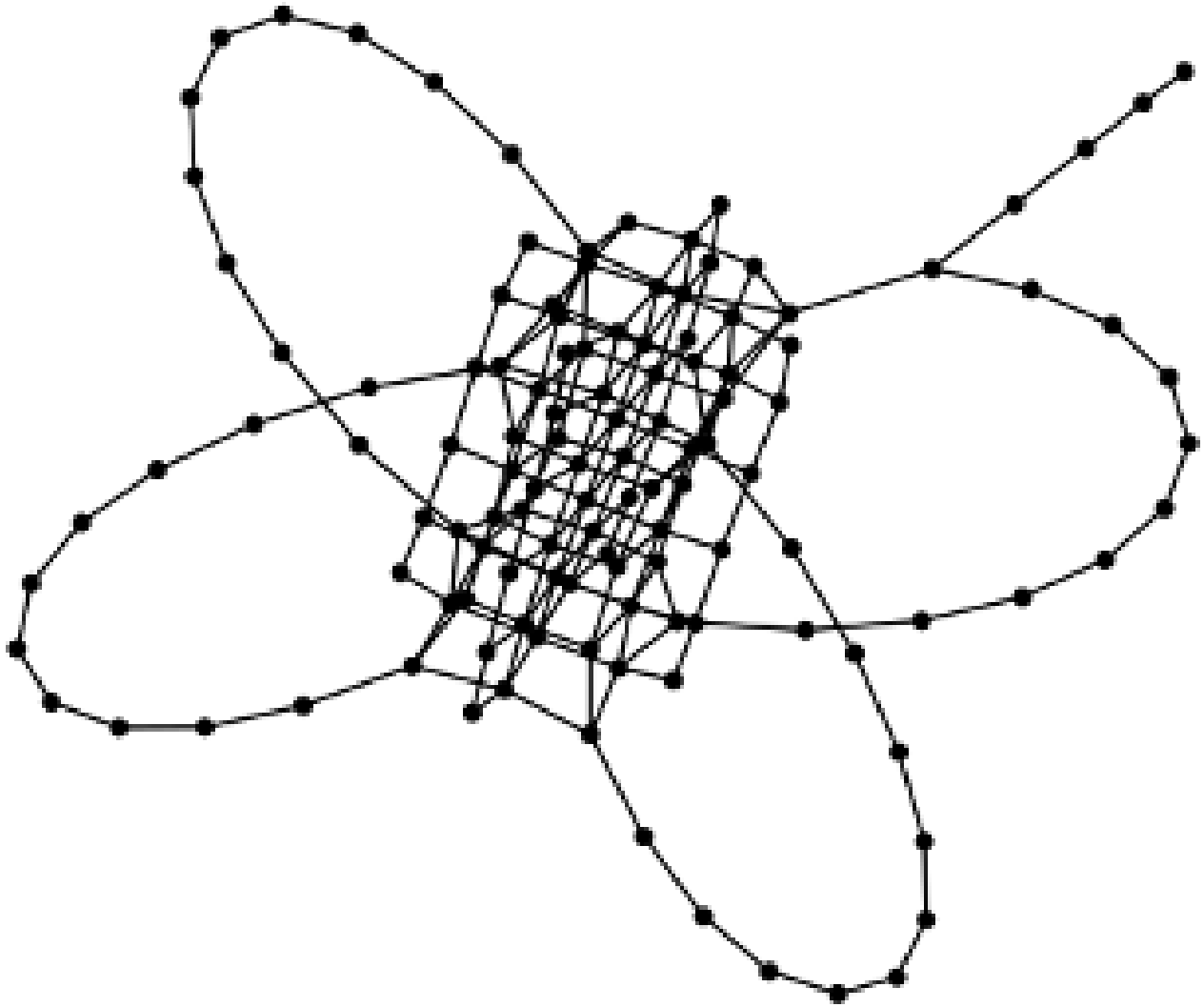
1. CUDA 6.5 и GPU с capability > 3 реально обеспечивают качественно новые возможности для программирования. Реализация динамического распараллеливания при таких возможностях не является сложной задачей.
2. Ситуация с паролями сегодня критическая, как никогда! Злоумышленник при помощи новых видеоадаптеров и алгоритмов может в течении короткого времени обращать криптостойкие хэш-функции.
3. Необходимо СРОЧНО проводить модернизацию устаревших криптографических схем, используемых для работы с паролями.

Разреженные матрицы и малоизвестные полезные матричные соотношения

$$\begin{pmatrix} L_1 & \\ L_{12} & L_2 \end{pmatrix} \begin{pmatrix} U_1 & U_{12} \\ & U_2 \end{pmatrix}^{-1} = \begin{pmatrix} U_1^{-1}L_1^{-1} + U_1^{-1}U_{12}U_2^{-1}L_2^{-1}L_{12}L_1^{-1} & -U_1^{-1}U_{12}U_2^{-1}L_2^{-1} \\ -U_2^{-1}L_2^{-1}L_{12}L_1^{-1} & U_2^{-1}L_2^{-1} \end{pmatrix}$$

Используется для рекурсивного LU-разложения

Рабочая разреженная матрица (АЭС с ВВЭР)



Результаты ускорения

Date	Solver method	Time
2012	Upper relaxation	9000
2012a	New opt. compiler	4800
2012b	UMFPACK, direct	530
2012c	SMatrix, adaptive	60-1500
2012d	ACML, direct	650
2013	Ch1(MSU), direct	100
2014	Ch1_FMA(MSU), direct	30

Стабилизация и оптимальное управление

Управление ядерной реакцией

Коэффициент размножения нейтронов

Характеризует быстроту роста числа нейтронов и равен отношению числа нейтронов в одном каком-либо поколении цепной реакции к породившему их числу нейтронов предшествующего поколения.

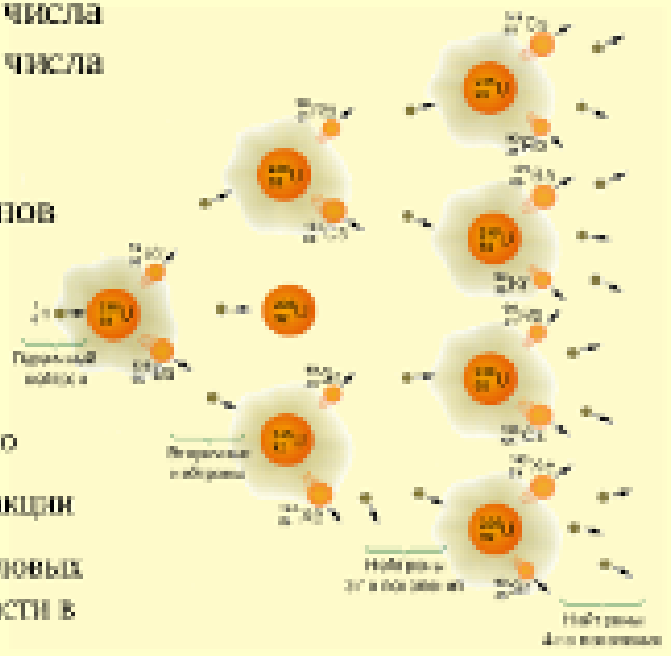
$$k = S_i / S_{i-1}$$

$k<1$ – Реакция затухает

$k=1$ – Реакция протекает стационарно

$k=1.006$ – Предел управляемости реакции

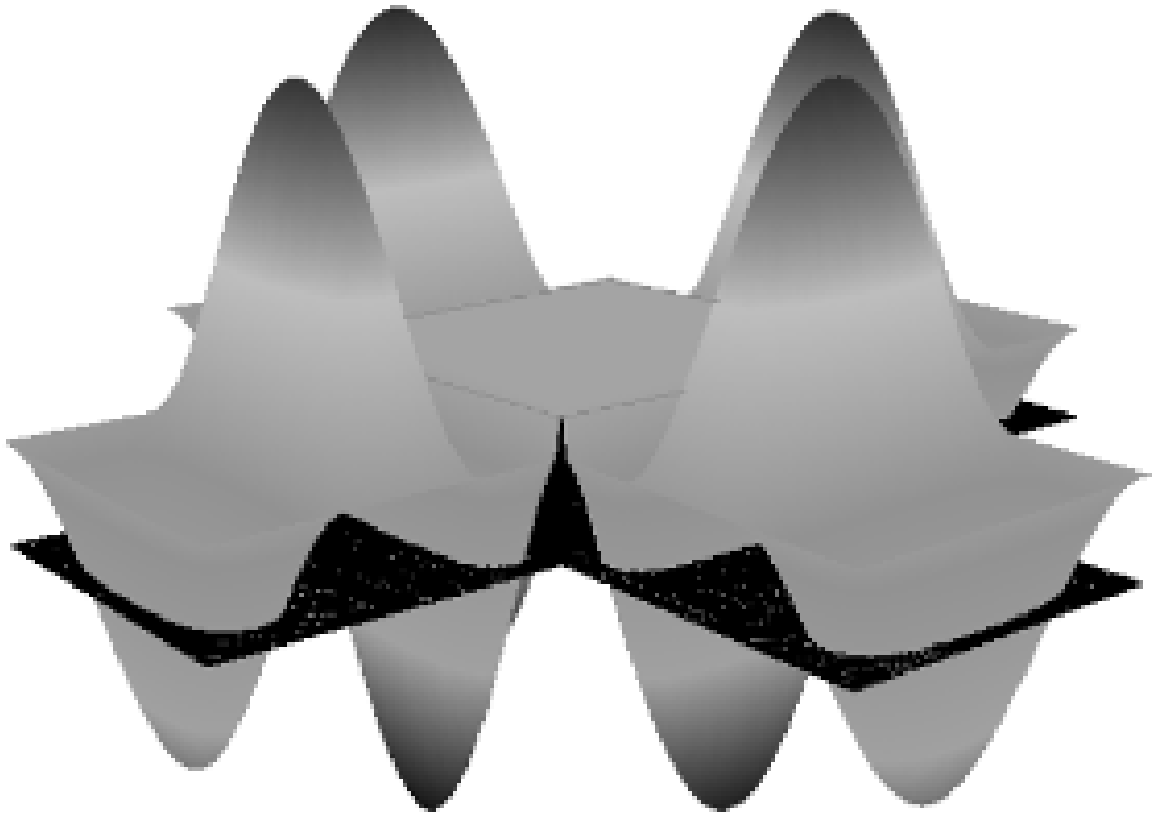
$k>1.01$ – Взрыв (для реактора на тепловых нейтронах энерговыделение будет расти в 20000 раз в секунду).



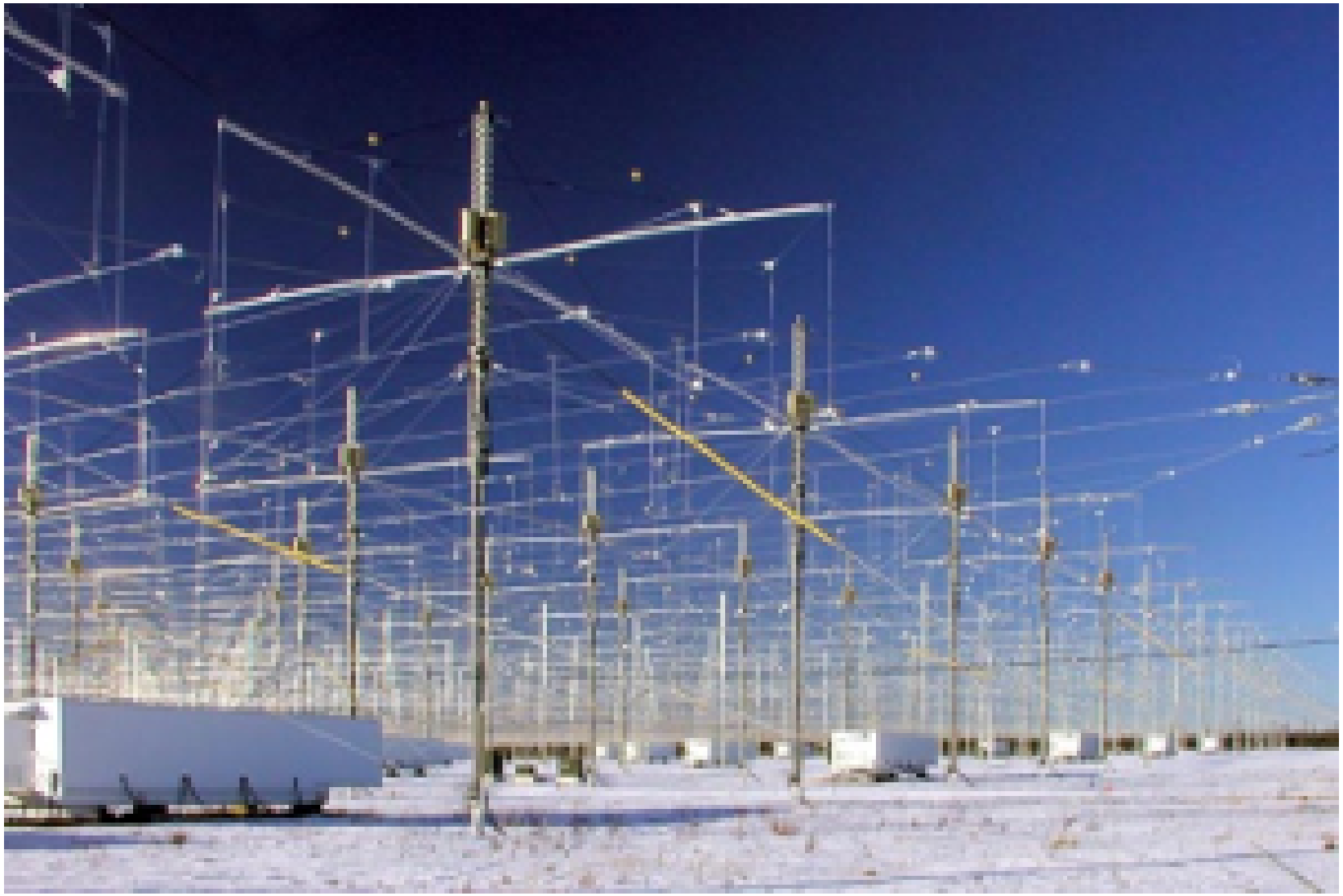
Типичный для урана ход цепной реакции; не показаны γ-кванты (~180 МЭВ) и нейтрино

От чего зависит коэффициент размножения нейтронов? 13

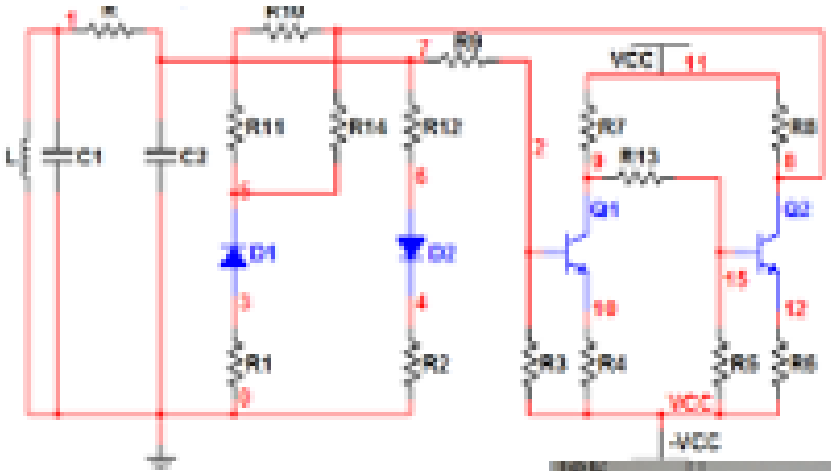
Оптимальное управление в технике

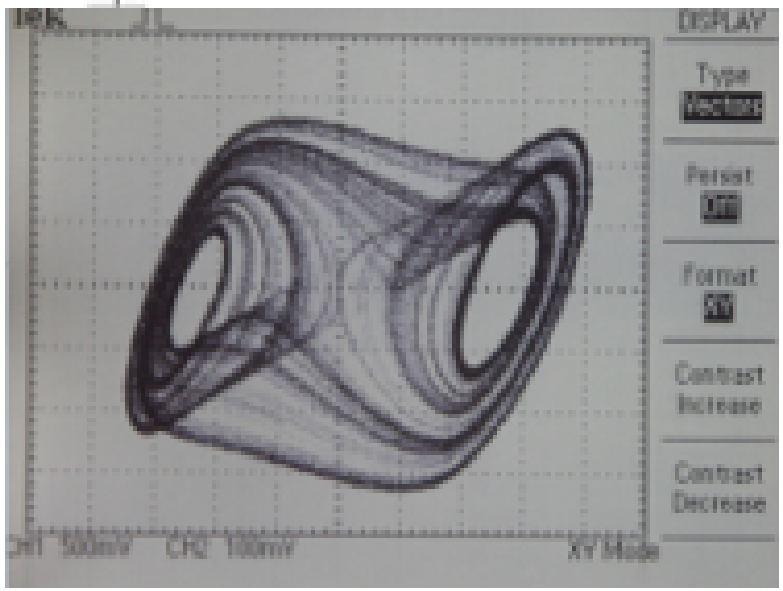


Климатическое оружие

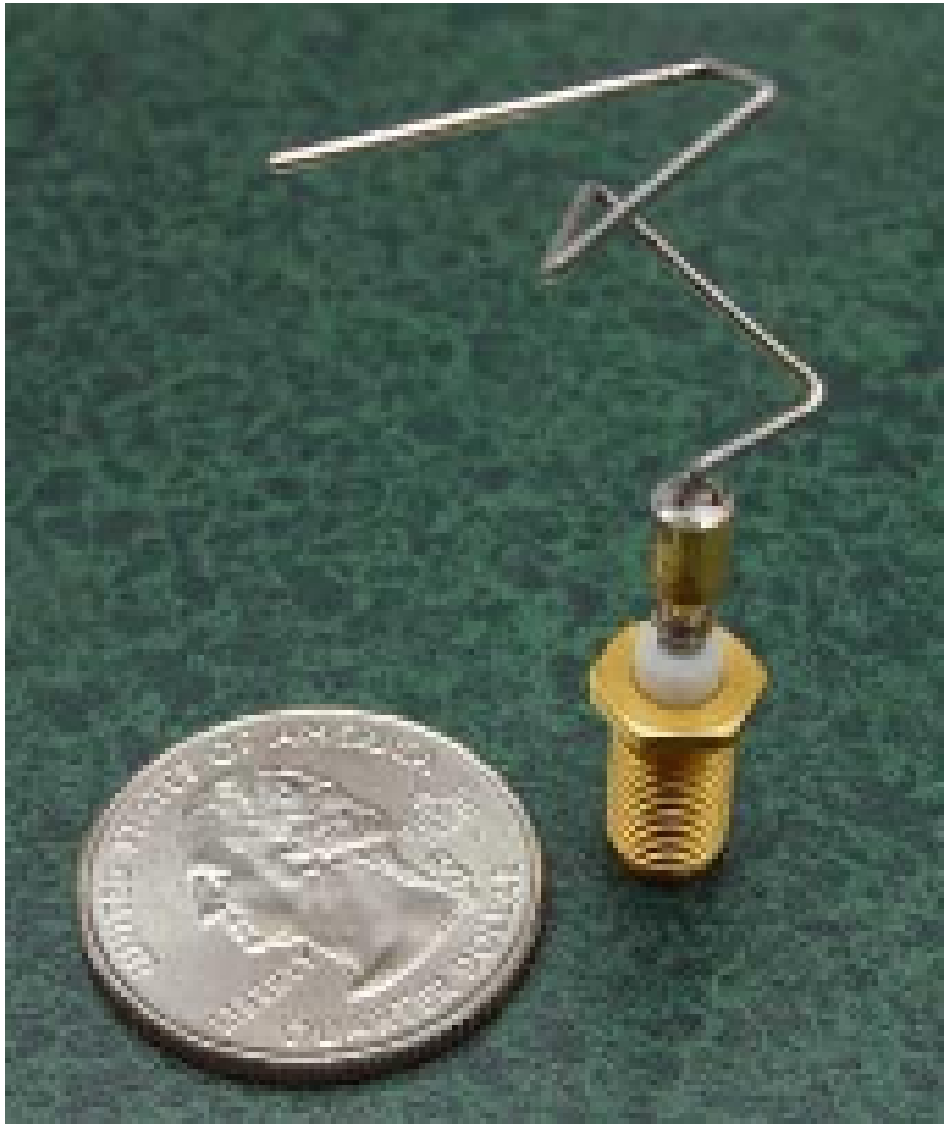


Аттракторы в простых нелинейных системах

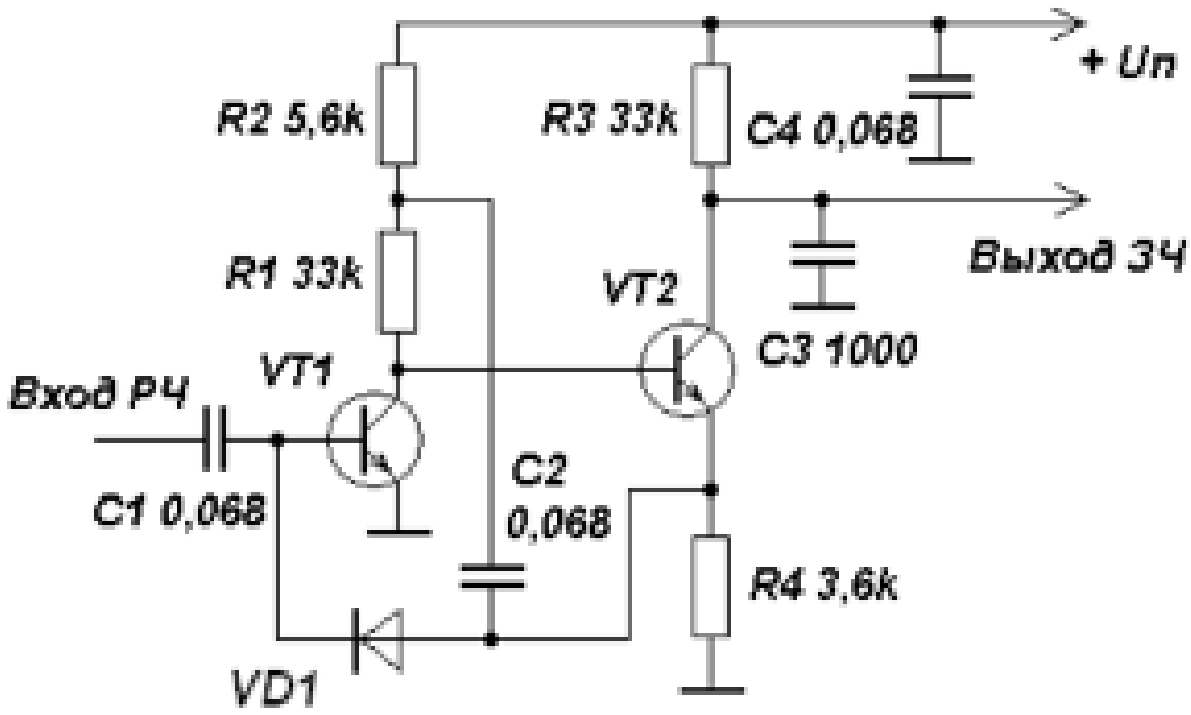




Генетические алгоритмы



Синтез/оптимизация АМ-детектора



Чувствительность в линейном режиме

```
Оригинальная схема: ~ 10^-8  Вт
Синтезированная:    ~ 10^-14 Вт
```

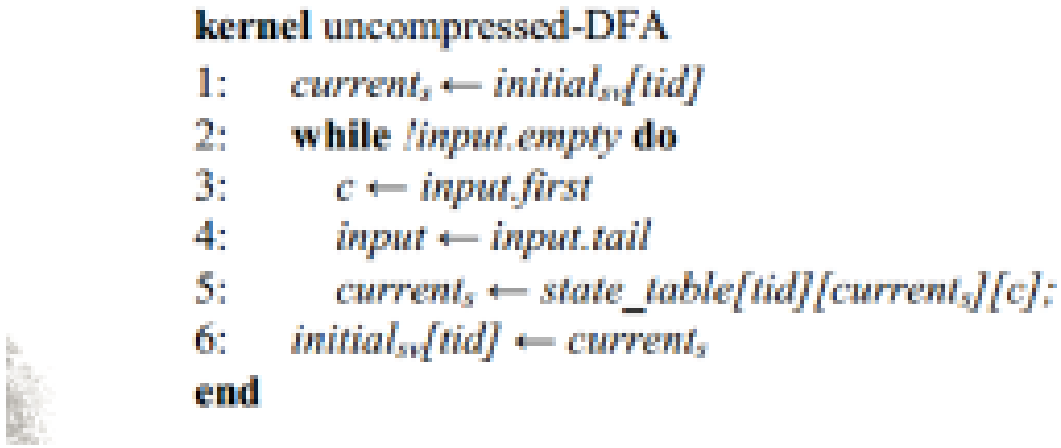
HPC-scanner

iNFAnt represents a significant departure from traditional software-based pattern matching engines both for its underlying automaton model, the NFA, and its target hardware platform, the GPU.

iNFAnt also represents, as far as we know, one of the first approaches to pattern matching designed from the ground up for the heavily parallel execution environment ordered of modern programmable GPUs, as opposed to being an adaptation of a technique originally designed for general-purpose processors.

Детерминированный конечный автомат

- Uncompressed DFA-based solution
  - DFAs are laid out next to one another in global memory
  - Current active state in a local register



```
kernel uncompressed-DFA
1:   current_s ← initial_sv[tid]
2:   while !input.empty do
3:       c ← input.first
4:       input ← input.tail
5:       current_s ← state_table[tid][current_s][c];
6:   initial_sv[tid] ← current_s
end
```

Сравнение производительности CPU vs GPU

Dataset	CPU		GPU				
	NFA	DFA	NFA	O-NFA	U-DFA	C-DFA	E-DFA
	P_M=0.35						
Backdoor	0.4	7.9	40.8	37.6	171.4	13.8	42.6
Spyware	0.9	4.0	40.1	46.4	168.2	11.5	31.8
E-M	3.9	40.3	14.3	27.3	235.6	-	-
Range.5	4.9	40.6	13.6	26.7	227.1	-	-
Range1	4.9	41.1	18.6	25.7	211.8	-	-
Dotstar.05	1.4	7.9	20.0	25.8	190.6	13.7	37.1
Dotstar.1	0.9	5.2	18.7	26.0	158.1	8.8	30.2
Dotstar.2	0.6	3.2	18.6	24.0	-	6.2	26.3
	P_M=0.75						
Backdoor	0.4	7.4	37.2	31.9	166.8	13.7	40.3
Spyware	0.4	4.3	35.3	35.9	168.2	9.3	32.5
E-M	3.1	42.4	10.0	24.9	218.4	-	-
Range.5	3.3	42.3	16.9	24.2	208.3	-	-
Range1	3.3	42.0	13.3	23.3	209.1	-	-
Dotstar.05	0.5	7.6	13.2	17.6	150.8	13.4	38.8
Dotstar.1	0.6	5.2	16.4	20.8	156.1	8.1	29.1
Dotstar.2	0.8	3.0	14.6	19.4	-	6.0	26.3

Некоторые выводы по данной задаче

- A comprehensive study of regular expression matching on GPUs
  - Datasets of practical size and complexity
  - Explored advantages and limitations of different NFA- and DFA-based representations
- Uncompressed DFA solution outperforms other implementations
  - On large and complex datasets exceeding the memory capacity
  - Schemes to improve a basic default-transition compressed DFA design

Высказывания “прорицателей” в HPC-сфере

- Через несколько лет появятся...
- Системы типа XXX обгонят системы типа YYY...
- СуперЭВМ XXX не подходит для...
- CUDA плохо подходит для...
- Метод XXX не годится для...
- Для того, чтобы посчитать XXX, нужно посчитать YYY...

Ребята, а ВЫ МОЖЕТЕ ЭТО ДАКАЗАТЬ?

Или доказательство в студию, или к Ванге !

Древняя и современная [HPC-]мудрость

Из жанра “однажды в китае”

- Как же ты в одолел столько бандитов?
- У меня была с собой волшебная палочка.
- Волшебная?! А как она выглядит?
- Как лом, только с ножом на конце.

Оружие не заменяет мозги, а лишь дополняет их.

Из жанра HPC

Чтобы решать сложную задачу из области HPC, часто недостаточно иметь только высокопроизводительную платформу; нужно также использовать искусный метод решения.

Десяток артефактов по теме

- BAGIRA / Конференция+статья в ANS
- Метод алгебраической просветки (ИМ СО РАН)
- Попадание в CUDA Альманах
- Свидетельство на программу для ЭВМ №2014612693
- Доклад в ИМ СО РАН (БАГИРА)
- Статья в Иркутском сборнике и доклад про конструкцию из CQ-QRP
- Доклад на конференции в Тарусе
- Статья ‘T++ для CUDA-устройств …’
- Отчет НИИ Механики МГУ (3D-теплогидродинамика)
- Сегодняшний доклад в МГУ

Учебный курс HPC для МГИУ также был дополнен соответствующим материалом.

Возможные точки роста

1. Публикации в журналах, участие в научных конференциях.
   Акцент следует делать на том, как изящно теперь решаются практически важные задачи.
2. Инновационные фонды и гранты (эффективнее после публикаций).
   Инновационные формы лучше подходят, так как у нас сегодня преобладает инновационная составляющая.

Много интересных задач и направлений

1. Многоаспектная оптимизация программ
2. Сверхбыстрые решатели для матриц (ИМ СО РАН)
3. Моделирование и реинжиниринг кода (Рос?Атом)
4. Задачи управления и стабилизации (Корнев,Фурсиков)
5. Многоагентные системы (ЦЭМИ)
   - Агенты как форма реализации адаптивной логики
     - Разного рода логистические задачи
6. Дальнейшие эксперименты на GTX 750 & Tesla K20
7. Гибридные вычислители (вкл.цифро-аналог)
8. Мемристорные супервычислительные системы

Импортозамещение

Используя эффективный решатель, можно обойтись менее мощным, полностью отечественным оборудованием. Гибридизация вычислительных методов позволяет эффективно использовать гибридное оборудование, состоящее не только из традиционного цифрового.

Примеры:

1. “Отечественные” CPU семейств MIPS, Sparc, Байкал смогут решать задачи с такой же скоростью, что и новейшие Intel Core i7.
2. Гибридное CPU+FPGA оборудование уже давно признано в HPC-кругах как эффективная связка для решения широкого круга задач.

Выводы

- Ориентация на практические задачи и новое оборудование дает интересные результаты.
- При решении реальных задач наши новые методы уже позволяют “догнать и перегнать” :)
- Наши Российско-Интернациональные традиции HPC живы и активно развиваются.

СПАСИБО за Внимание!

Вопросы?