ВМК МГУ

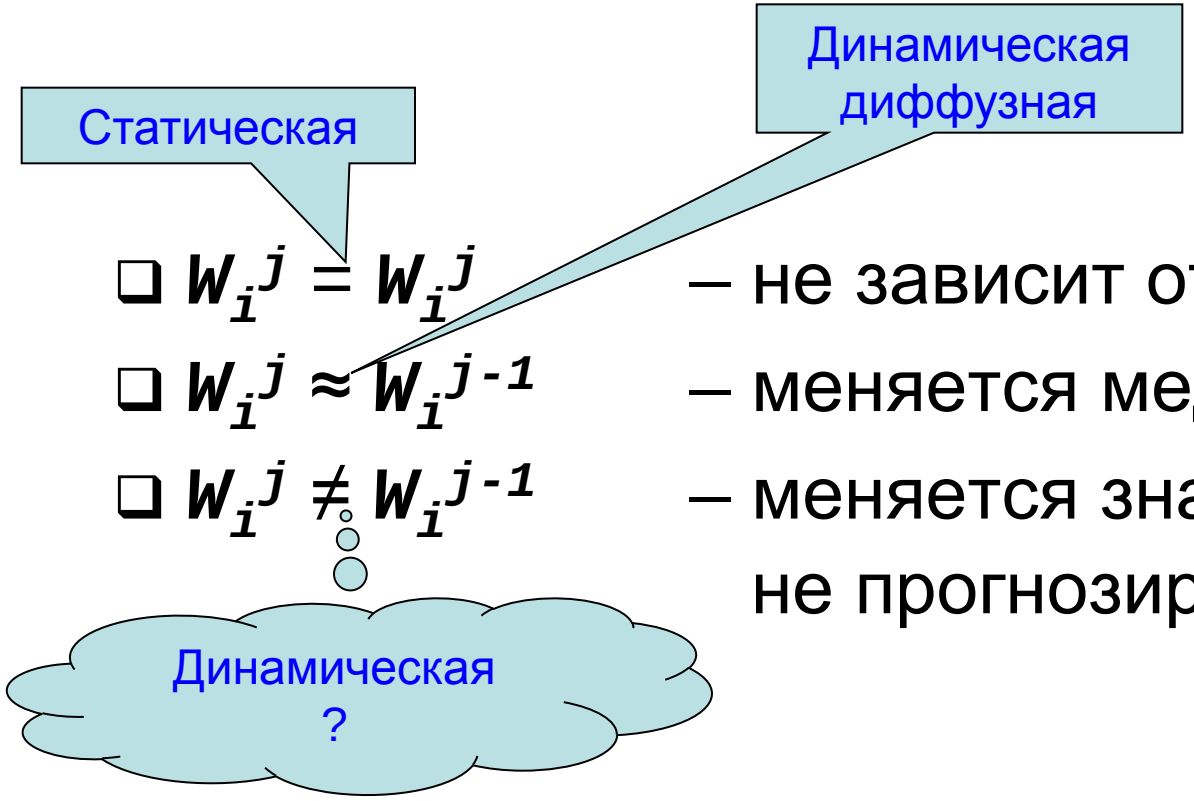
Динамическая балансировка загрузки процессоров

Параллельные вычисления

Якобовский Михаил Владимирович

Стратегии балансировки загрузки

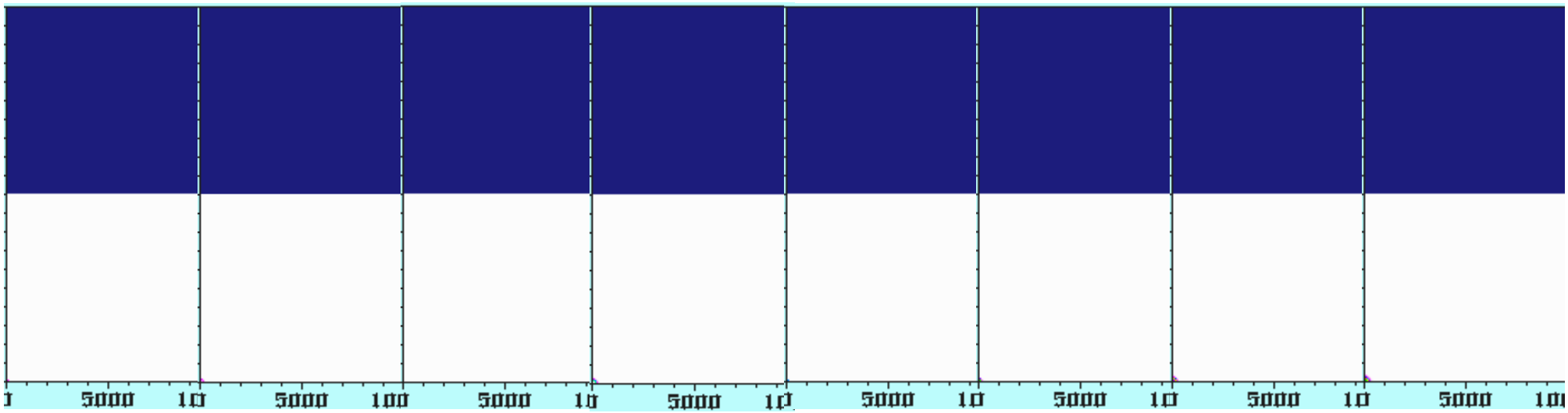
W_i^j - вычислительная нагрузка, ассоциированная с узлом сетки i на шаге j



- $W_i^j = W_i^j$ – не зависит от времени
- $W_i^j \approx W_i^{j-1}$ – меняется медленно
- $W_i^j \neq W_i^{j-1}$ – меняется значительно и не прогнозируемо

Methane combustion

CH_4 NO NO_2 N_2O CH_3 CO CO_2 T



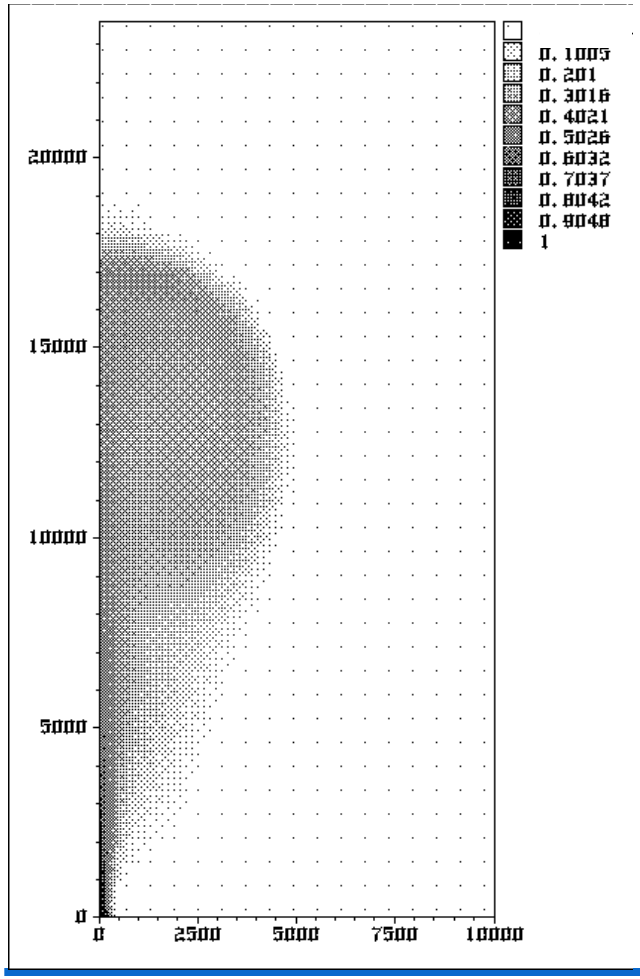
N_2, CH_4, CH_3, CH_3O, CH_2O, CHO, CO, CO_2, H_2, O_2, H, OH, HO_2, H_2O.

13 веществ - 45 реакций

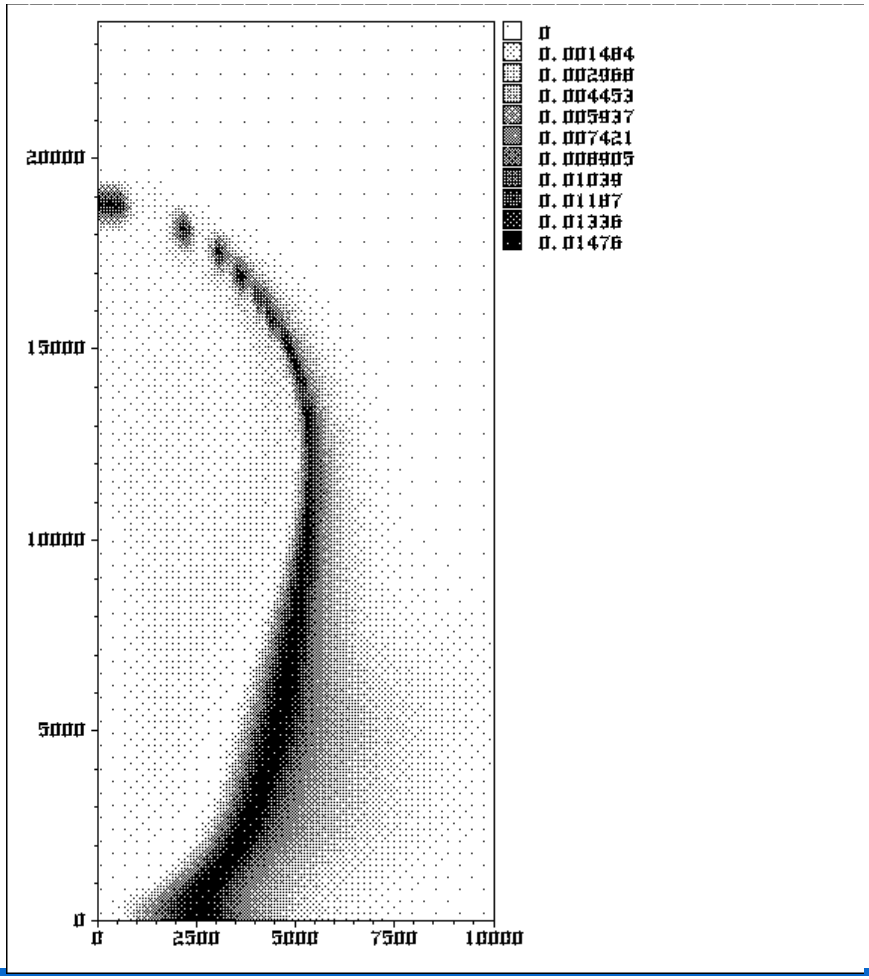
CH_4, CH_3, CH_2O, CH_3O, CHO, CO, CO_2, H_2, O_2, H, OH, HO_2, H_2O, N_2, N, NO, NO_2, N_2O.

19 веществ - 64 реакции

CH4


0.1005
0.201
0.3016
0.4021
0.5026
0.6032
0.7037
0.8042
0.9048
1
20000
15000
10000
5000
0
0
2500
5000
7500
10000

NO


0
0.001484
0.002968
0.004453
0.005937
0.007421
0.008905
0.01039
0.01187
0.01336
0.01476
20000
15000
10000
5000
0
0
2500
5000
7500
10000

$$\frac{\partial \rho}{\partial t}+\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{\partial y}(\rho v)=\frac{\partial}{\partial x}\tau\frac{\partial}{\partial x}(\rho u^2+p)+\frac{\partial}{\partial y}\tau\frac{\partial}{\partial y}(\rho v^2+p), \quad (1)$$

$$\frac{\partial \rho y^{(i)}}{\partial t}+\frac{\partial}{\partial x}(\rho y^{(i)}u)+\frac{\partial}{\partial y}(\rho y^{(i)}v)=\frac{\partial}{\partial x}\tau\frac{\partial}{\partial x}(\rho y^{(i)}u^2+p^{(i)})+\frac{\partial}{\partial y}\tau\frac{\partial}{\partial y}(\rho y^{(i)}v^2+p^{(i)})+\omega^{(i)}, \quad (2)$$

$$\frac{\partial \rho u}{\partial t}+\frac{\partial}{\partial x}(\rho u^2+p)+\frac{\partial}{\partial y}(\rho uv)=\frac{\partial}{\partial x}\tau\frac{\partial}{\partial x}(\rho u^3+3pu)+\frac{\partial}{\partial y}\tau\frac{\partial}{\partial y}(\rho uv^2+pu), \quad (3)$$

$$\frac{\partial \rho v}{\partial t}+\frac{\partial}{\partial x}(\rho uv)+\frac{\partial}{\partial y}(\rho v^2+p)=\frac{\partial}{\partial x}\tau\frac{\partial}{\partial x}(\rho u^2v+pv)+\frac{\partial}{\partial y}\tau\frac{\partial}{\partial y}(\rho v^3+3pv), \quad (4)$$

$$\frac{\partial E}{\partial t}+\frac{\partial}{\partial x}(u(E+p))+\frac{\partial}{\partial y}(v(E+p))=\frac{\partial}{\partial x}\tau\frac{\partial}{\partial x}(u^2(E+2p))+$$

$$+\frac{\partial}{\partial y}\tau\frac{\partial}{\partial y}(v^2(E+2p))+\frac{\partial}{\partial x}\tau\frac{\partial\theta}{\partial x}+\frac{\partial}{\partial y}\tau\frac{\partial\theta}{\partial y}, \quad (5)$$

$$\theta=\sum_{(i)}p^{(i)}h^{(i)}+0.5p(u^2+v^2), \quad p=\sum_{(i)}p^{(i)}=\sum_{(i)}\rho^{(i)}R^{(i)}T, \quad \rho=\sum_{(i)}\rho^{(i)},$$

$$E=\rho(\varepsilon+0.5(u^2+v^2)), \quad R^{(i)}=R_0/m^{(i)}, \quad \varepsilon=\sum_{(i)}y^{(i)}\varepsilon^{(i)}, \quad y^{(i)}=\rho^{(i)}/\rho,$$

$$\varepsilon^{(i)}=\int_0^T c_V^{(i)}dT+h_o^{(i)}, \quad h^{(i)}=\int_0^T c_p^{(i)}dT+h_o^{(i)}, \quad \tau=\frac{\mu}{p}, \quad \frac{\mu}{\mu_0}=\left(\frac{T}{T_0}\right)^{\nu}, \text{ где } \nu=0.7,$$

$$\omega^{(i)}=m^{(i)}(Q_i-L_in_i) \qquad n_i=\frac{\rho}{m_i} \qquad c_p^{(i)}=\begin{cases}(R_0/m^{(i)})\cdot\sum_{j=1}^{5}b_{1j}^{(i)}T^{j-1}, & T\geq 1000\\ (R_0/m^{(i)})\cdot\sum_{j=1}^{5}b_{2j}^{(i)}T^{j-1}, & T\leq 1000\end{cases}$$

Начальные условия:

Воздух при $T_0 = 290$ °К $p_0 = 1$ атм, $u_0 = v_0 = 0$.

Граничные условия:

1) Нижняя граница (отверстие):

$T, \rho_0, y_0^{(i)}=0, y_0^{(CH4)}=1, u_0=0, v = v_1$.

2) Нижняя граница (кроме отверстия):

$u = v = 0$, а для остальных функций: $\frac{\partial f}{\partial n} = 0$

3) и 4) Внешние границы: $\frac{\partial f}{\partial n} = 0$

5) Ось симметрии: $u = 0$, а для остальных функций условие $\frac{\partial f}{\partial n} = 0$

Метод решения

Система записывается в операторном виде:

$\frac{\partial \mathbf{U}}{\partial t} + A\mathbf{U} = f,\ \mathbf{U} = \left(\rho, \rho y^{(i)}, \rho u, \rho v, E\right)^T$, A - нелинейный оператор, $f = (0, \omega_i, 0, 0, 0)^T$

Система расщепляется по физическим процессам на блоки **газовой динамики (I) и химической кинетики (II)**:

I. $\frac{\partial \mathbf{U}}{\partial t} + A\,\mathbf{U} = 0.$

Аппроксимируется с помощью полунеявной разностной схемы

$$\frac{\mathbf{U}^{j+1} - \mathbf{U}^{j}}{\Delta t} + \frac{1}{2}\left(A\,\mathbf{U}^{j+1} + A\,\mathbf{U}^{j}\right) = 0.$$

Решается итерационным методом

$$\frac{\overset{(k+1)}{\mathbf{U}}{}^{j+1} - \mathbf{U}^{j}}{\Delta t} + \frac{1}{2}\left(A\,\overset{(k)}{\mathbf{U}}{}^{j+1} + A\mathbf{U}^{j}\right) = 0, \qquad 0 \le k \le s-1; \quad \overset{(0)}{\mathbf{U}}{}^{j+1} = \mathbf{U}^{j}.$$

II. $\frac{d\mathbf{U}}{dt} = f,$ где $f = (0, \omega_i, 0, 0, 0)^T$.

На каждом временном интервале $(t, t+\Delta t)$ поочередно решаются блоки уравнений газовой динамики (ГД) и химической кинетики (ХК).

I Решаются уравнения (1)-(5) при $\omega^{(i)} = 0$

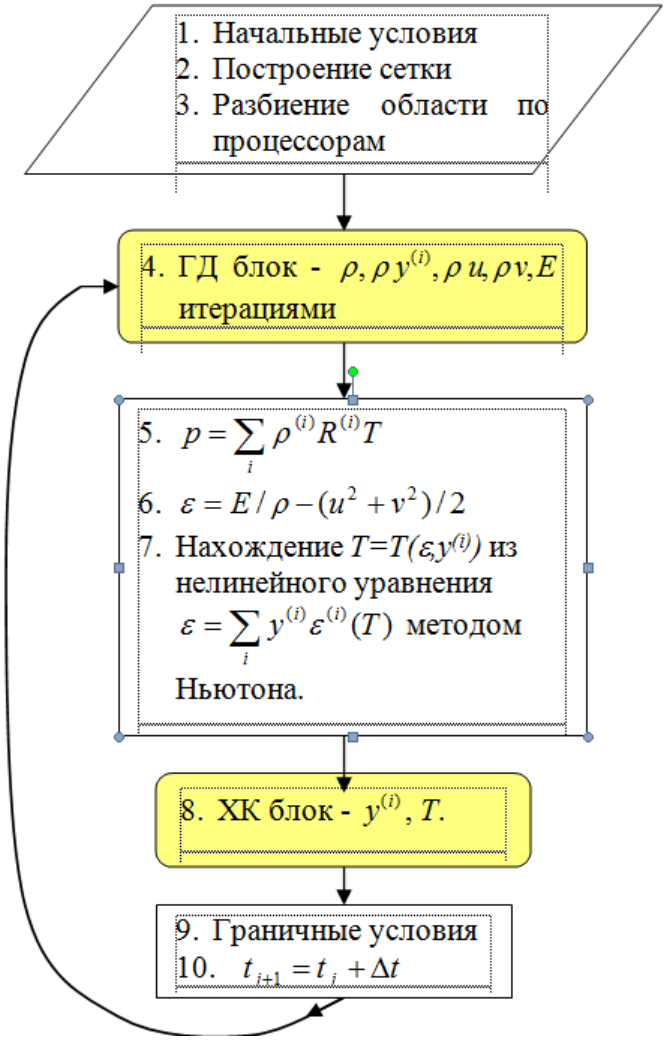
II Система (10) сводится к уравнению $\frac{d\rho^{(i)}}{dt} = \omega^{(i)}$, или для молярных концентраций:

$$\frac{dn^{(i)}}{dt} = Q^{(i)} - n^{(i)} L^{(i)}$$

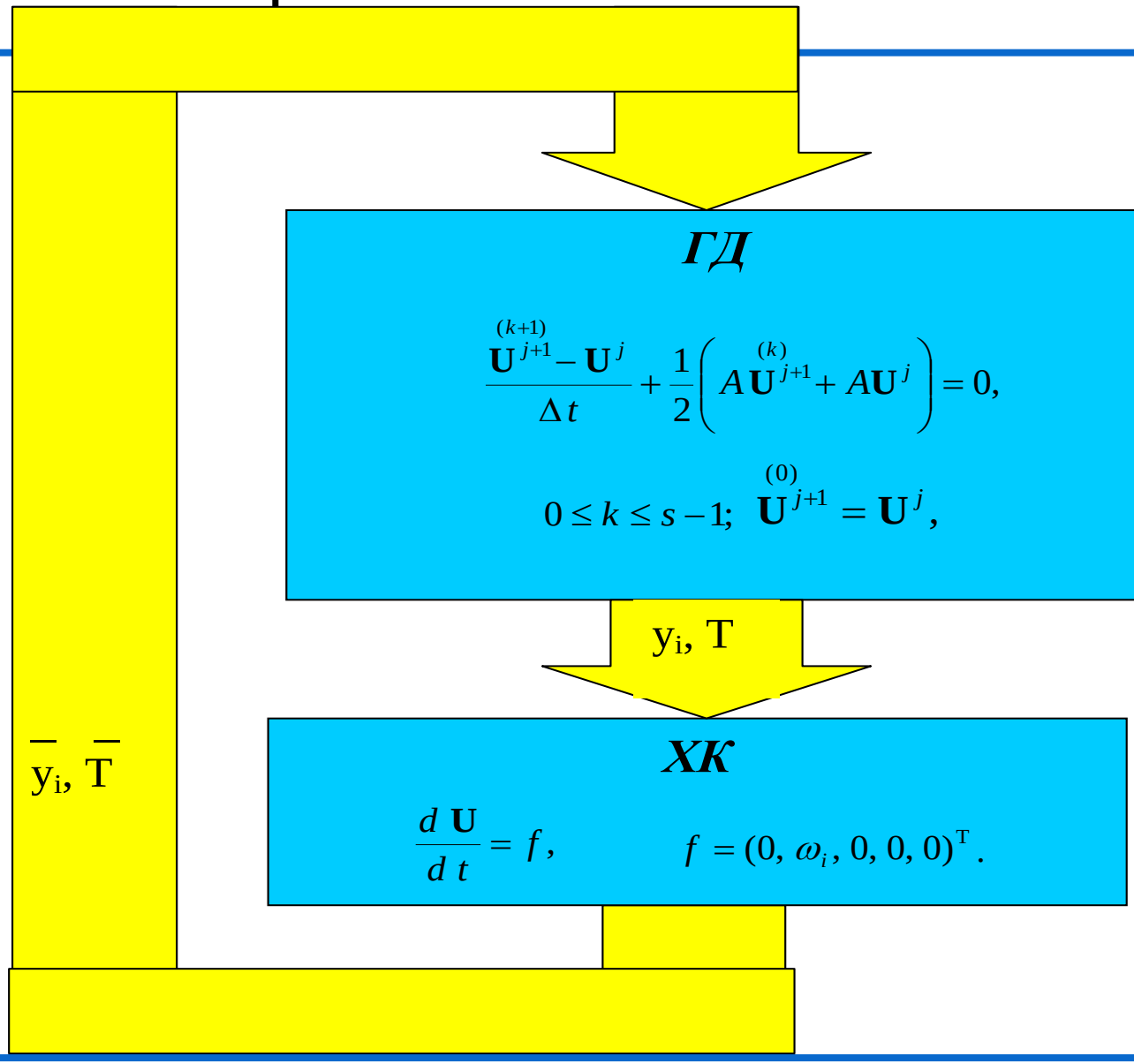
Изменение температуры за счет химических реакций

$$\frac{dT}{dt} = -\sum_{(i)} \omega^{(i)} \varepsilon^{(i)} / \sum_{(i)} \rho^{(i)} c_V^{(i)}$$

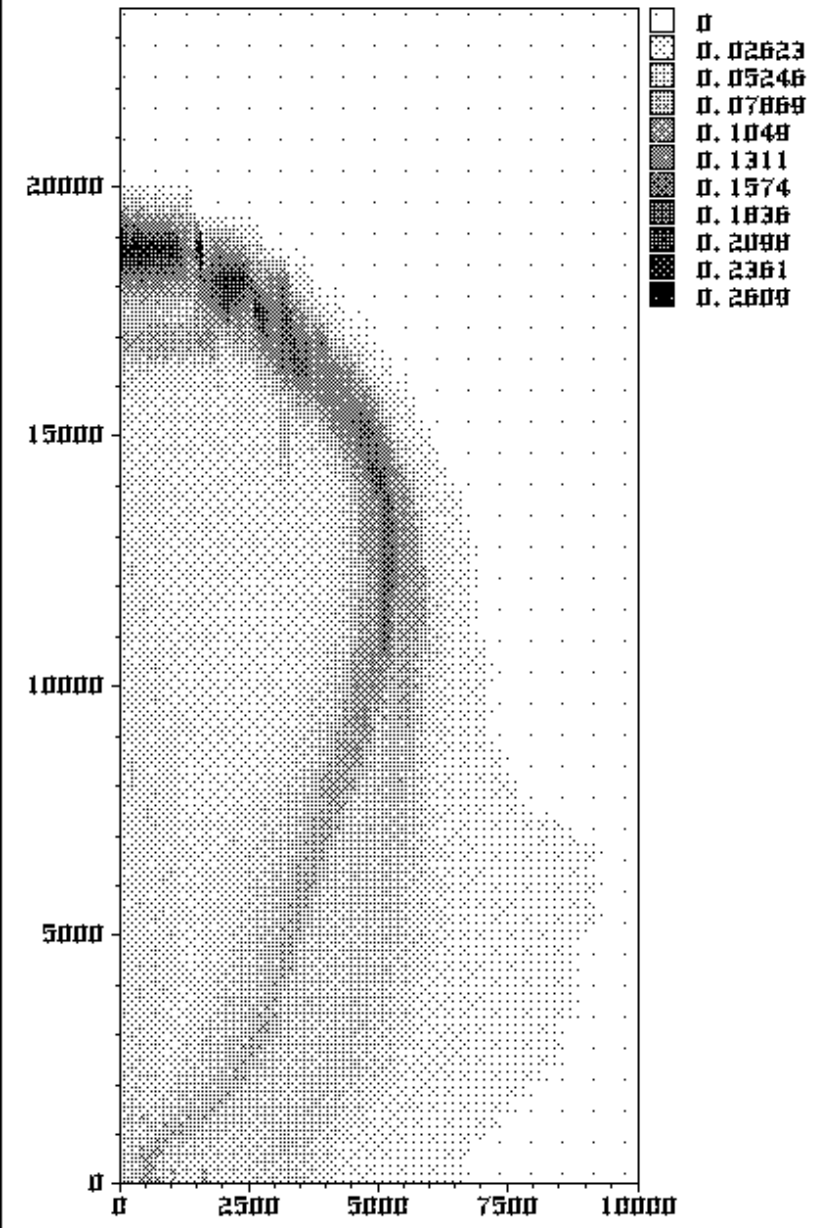
Блок схема вычислений



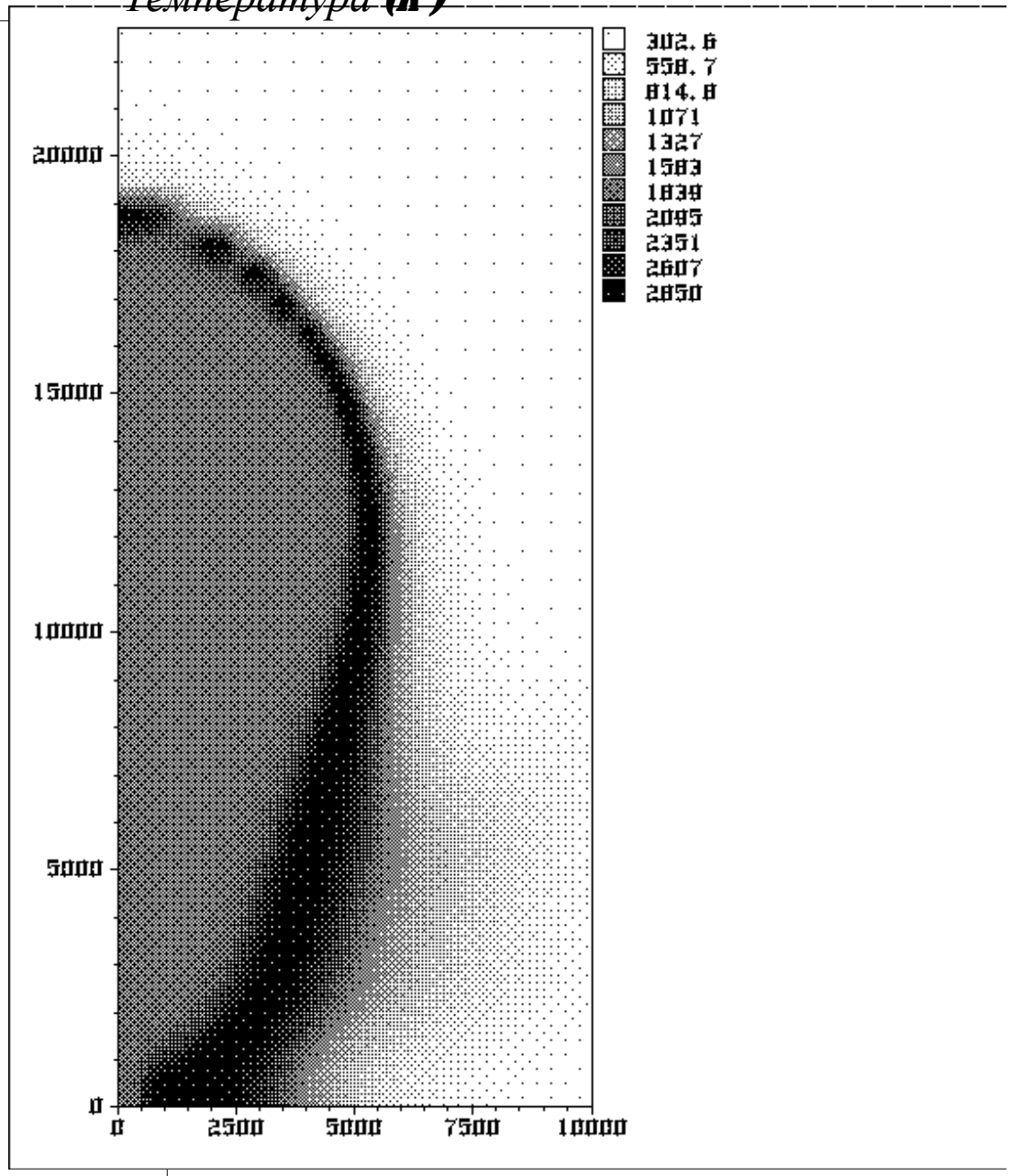
Блок схема алгоритма



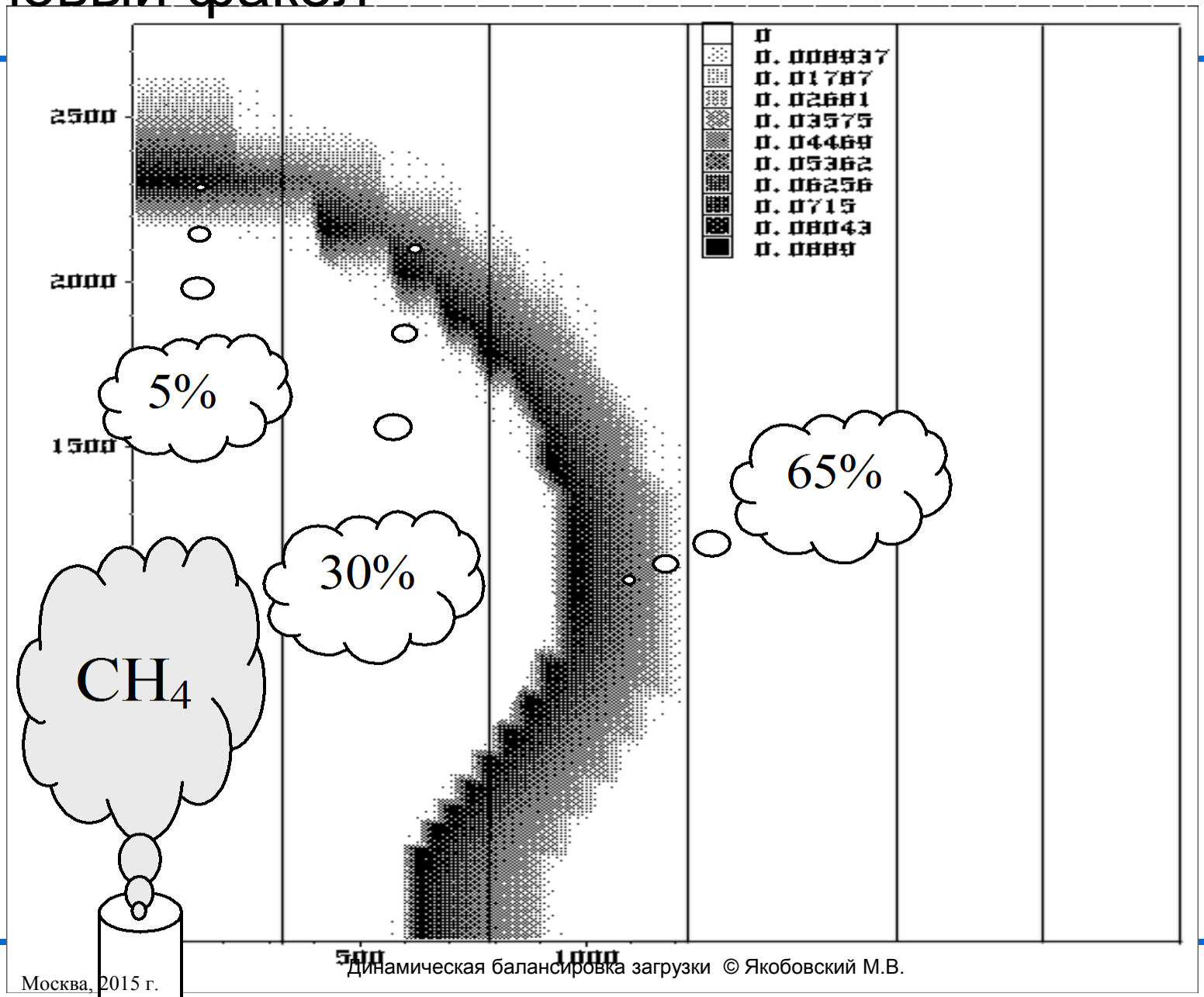
Время (сек)



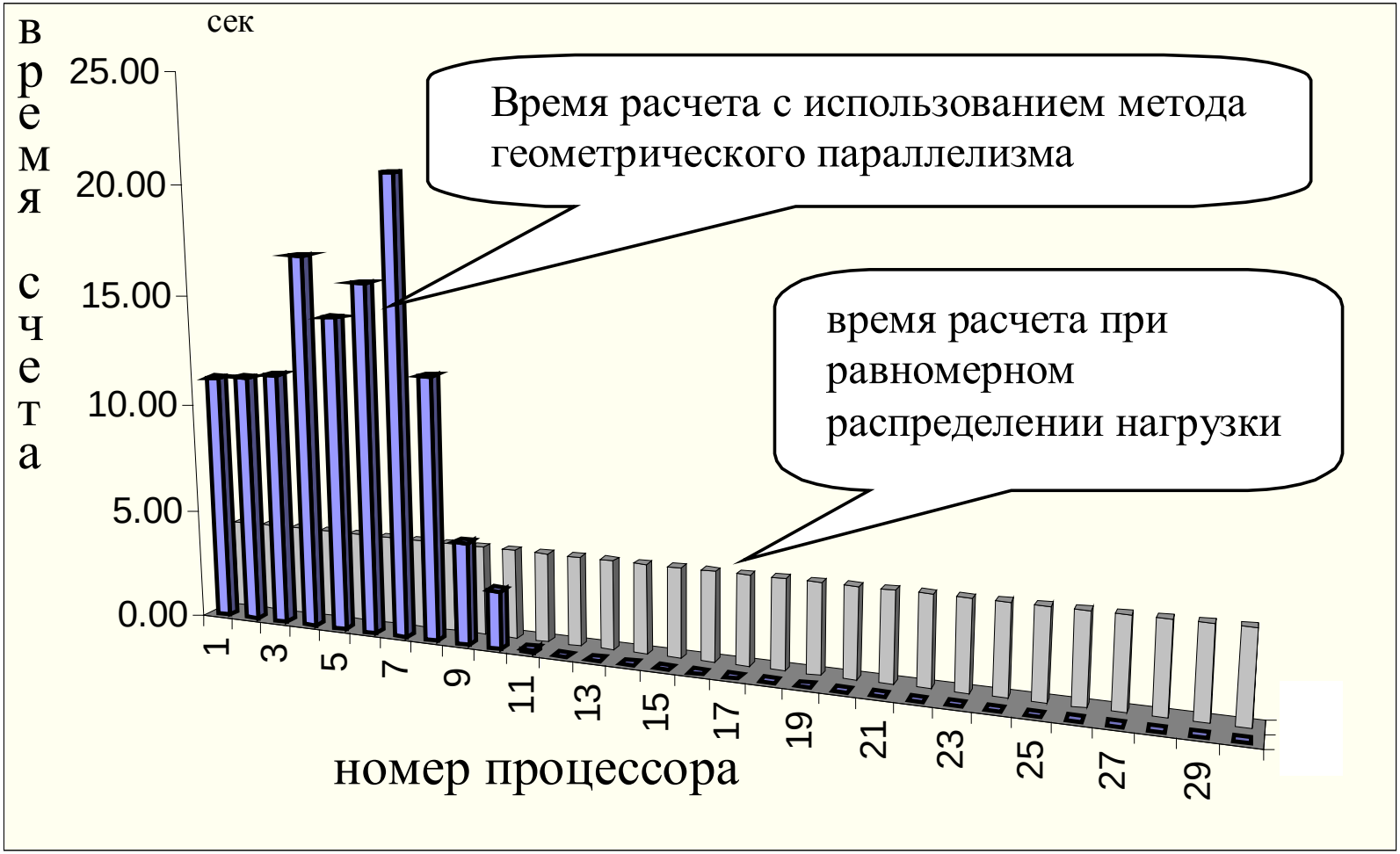
Температура (K°)



Метановый факел

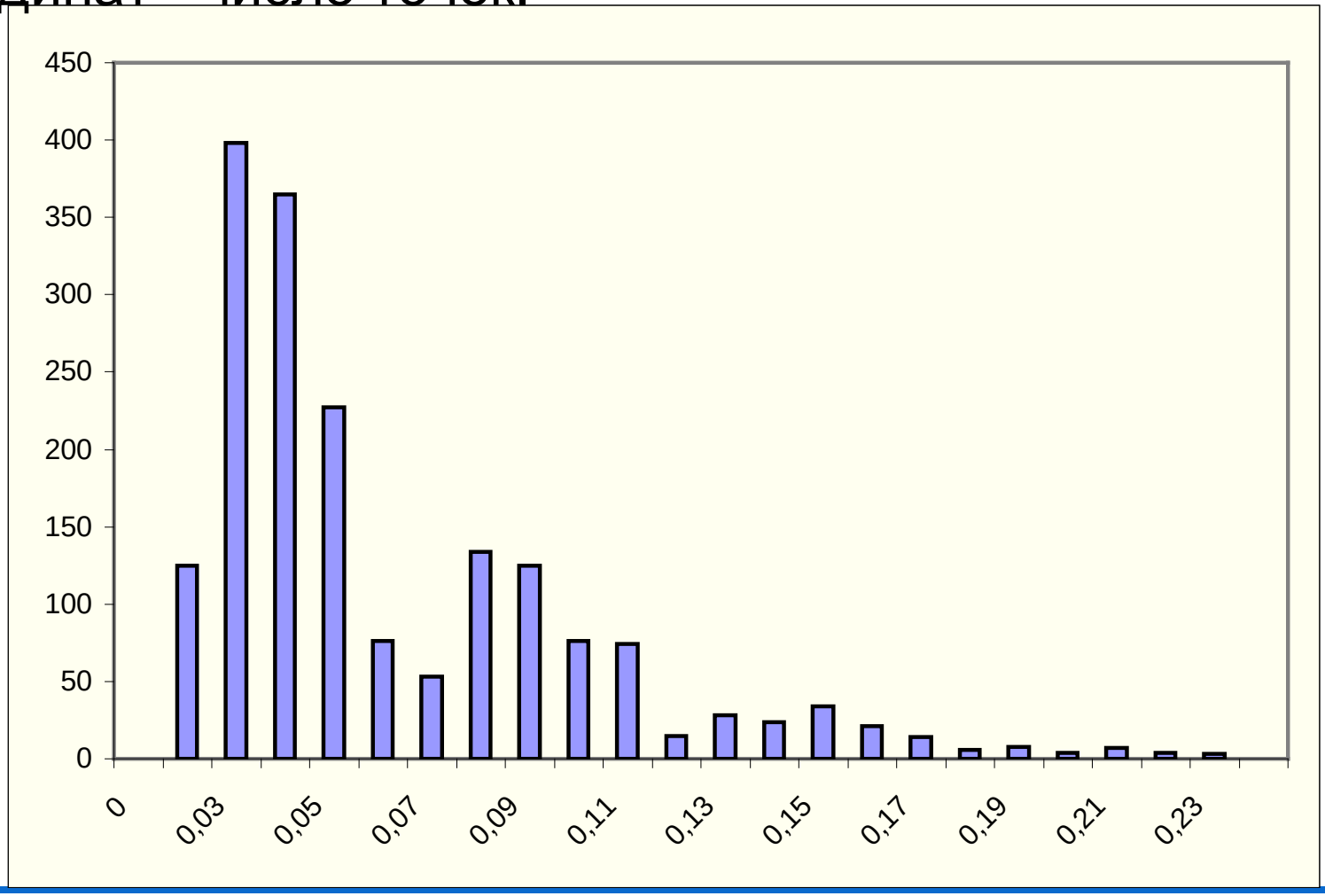


Распределение вычислительной нагрузки по процессорам



Число точек с соответствующим временем обработки при расчете блока химической кинетики

Ось абсцисс - время обработки одной точки (сек)
ось ординат - число точек.

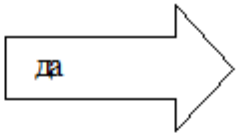


Основные положения

1. Отсутствует централизованный сбор данных управляющим процессором, как при коллективном решении
2. Отсутствует централизованное управление и сбор информации о наличии горячих точек у процессоров и текущей загрузке системы.
3. Преимущественная обработка локальных точек
4. Стратегия поиска работы
5. Коммуникации (обмен данными и сообщениями) на фоне обработки
6. Вторичное перераспределение точек
7. Возврат точек напрямую, минуя промежуточные этапы
8. Сообщения обрабатываются сразу же после получения
9. На каждый запрос поступает ответное сообщение
10. На каждом процессоре выполняются управляющий и обрабатывающий процессы

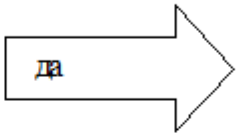
Цикл управляющего процесса

1. *Проверка условий возможности обработки точки.*



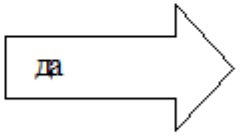
Начать обработку точки

2. *Проверка условий возможности посылки запроса на получение необработанных точек*



Послать запрос

3. *Проверка условий завершения работы*



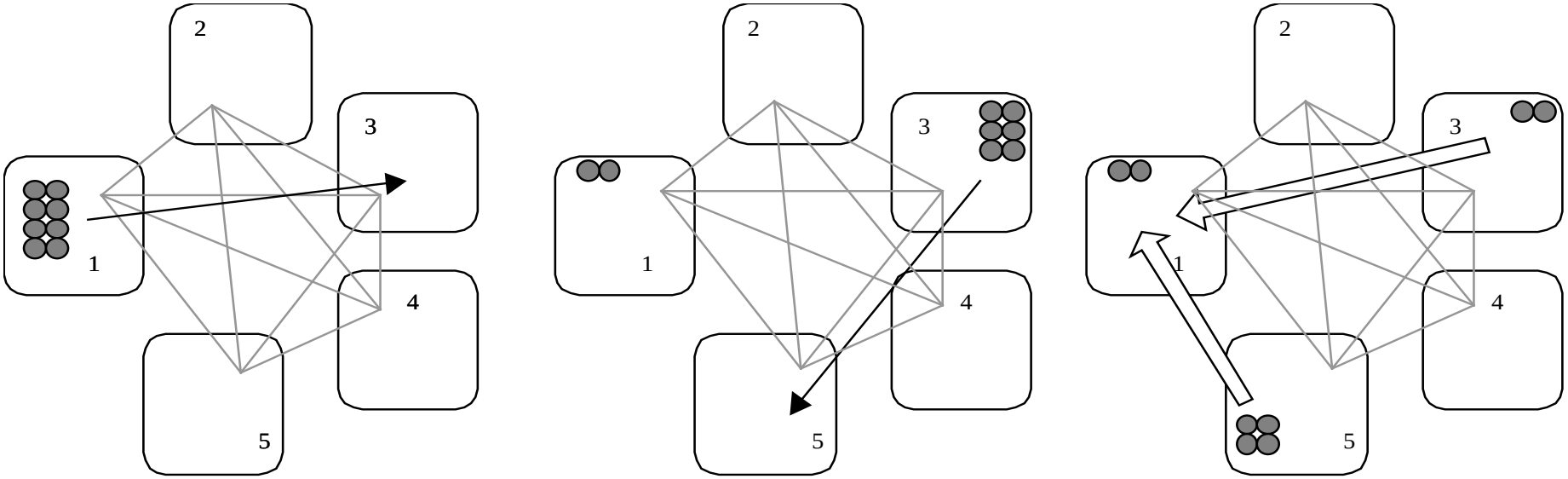
Завершить работу

4. *Получить очередное сообщение*

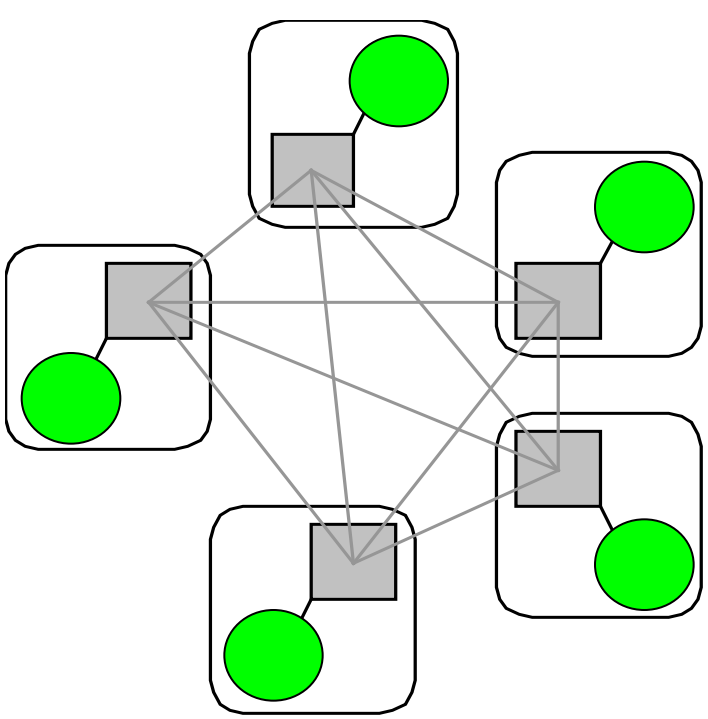
5. *Обработать сообщение*

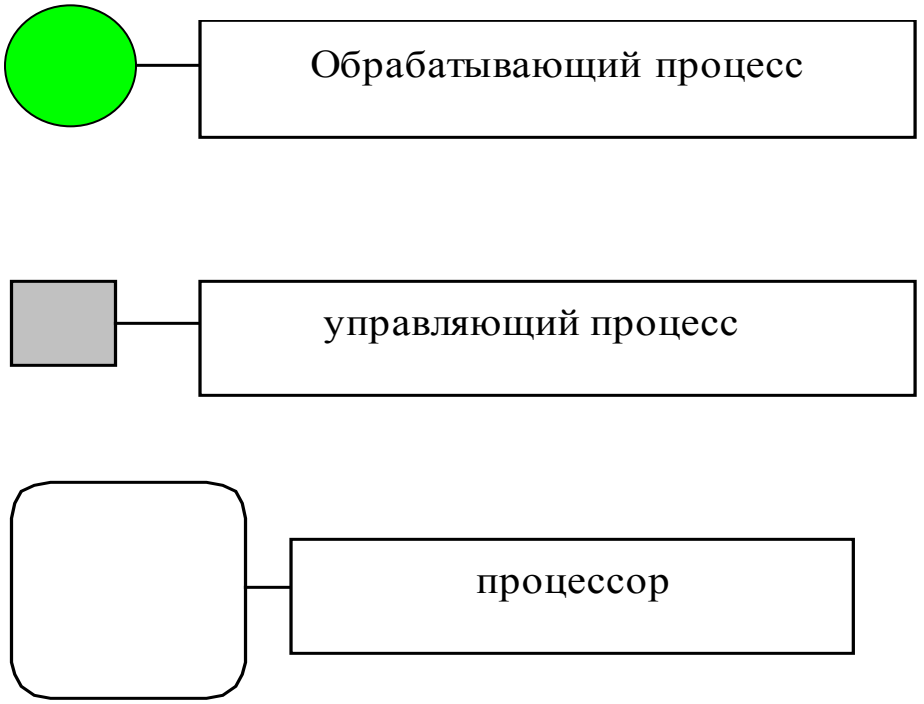
6. *Перейти к началу цикла*

Динамичекая балансировка



Основные процессы

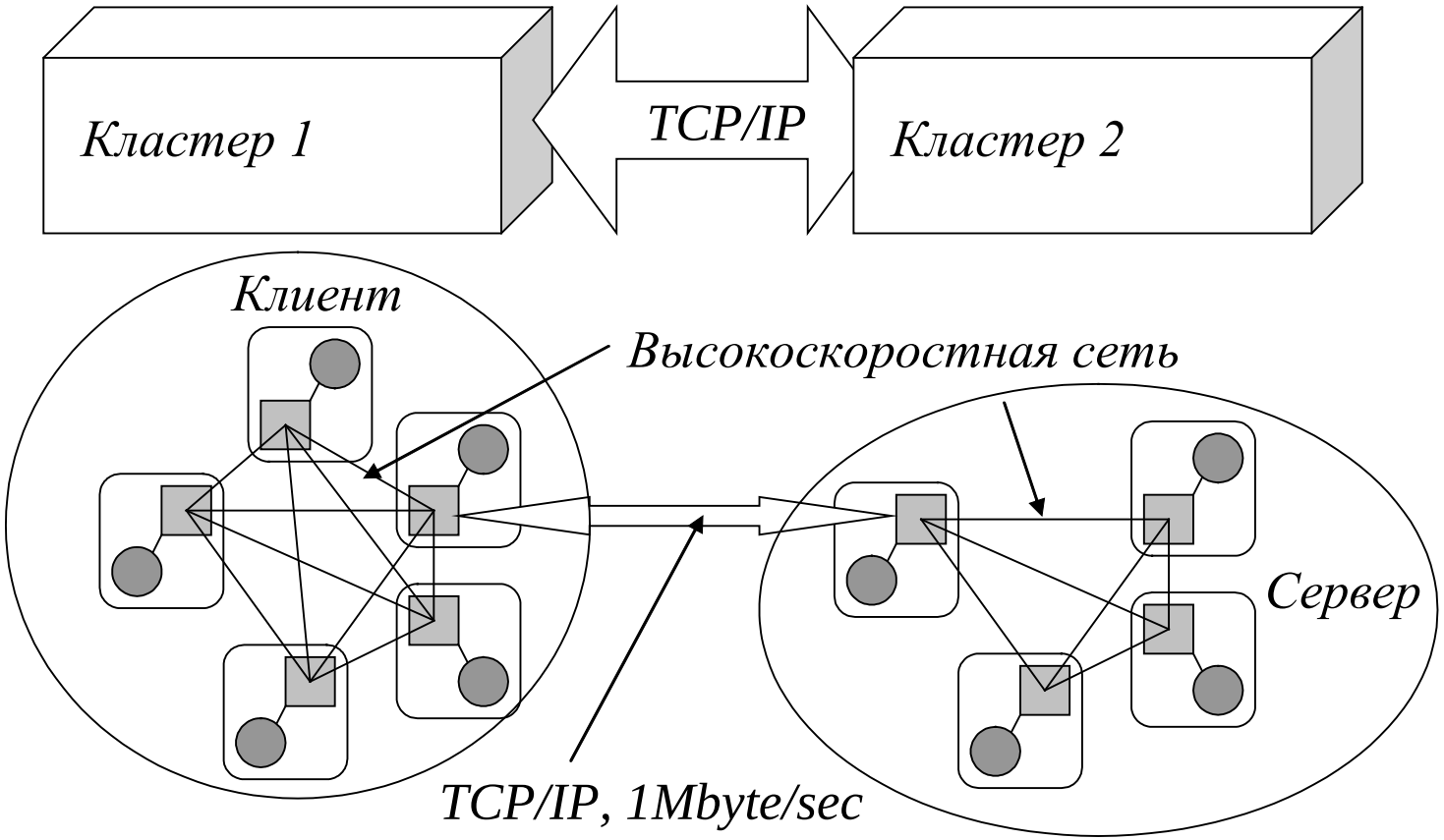




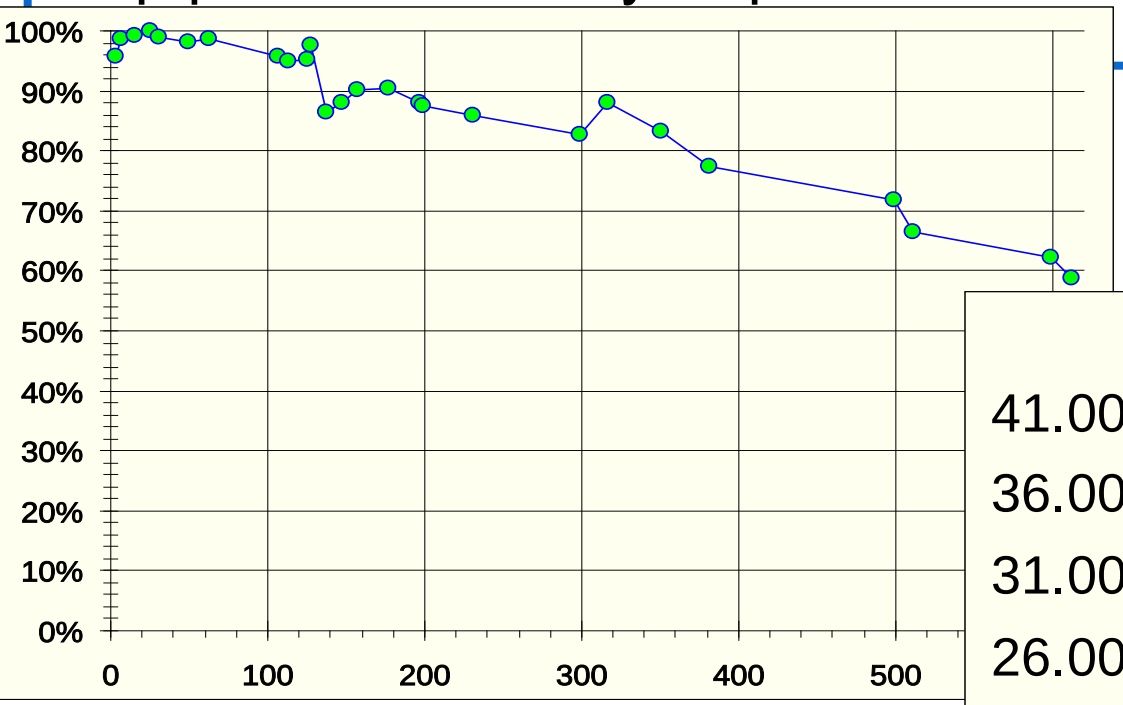
Завершение работы при выполнение всех условий

- нет локальных необработанных точек
- нет внешних точек
- нет обрабатываемых точек
- всем процессорам был послан запрос на получение необработанных точек
- всем процессорам было послано сообщение о том, что необработанные точки предоставлены быть не могут
- от всех процессоров получено сообщение о том, что необработанные точки предоставлены быть не могут
- все локальные точки обработаны и получены результаты обработки всех переданных точек

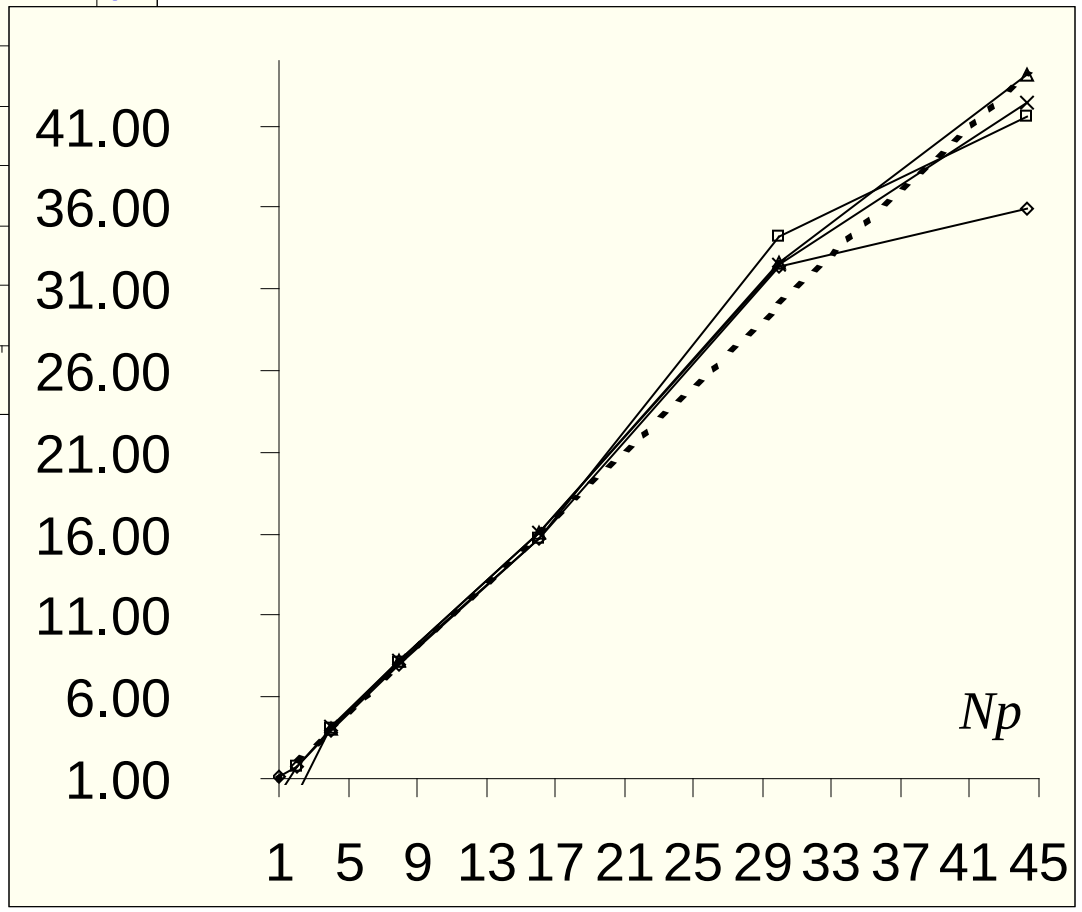
Структура программы при совместном использовании двух многопроцессорных комплексов



Эффективность и ускорение

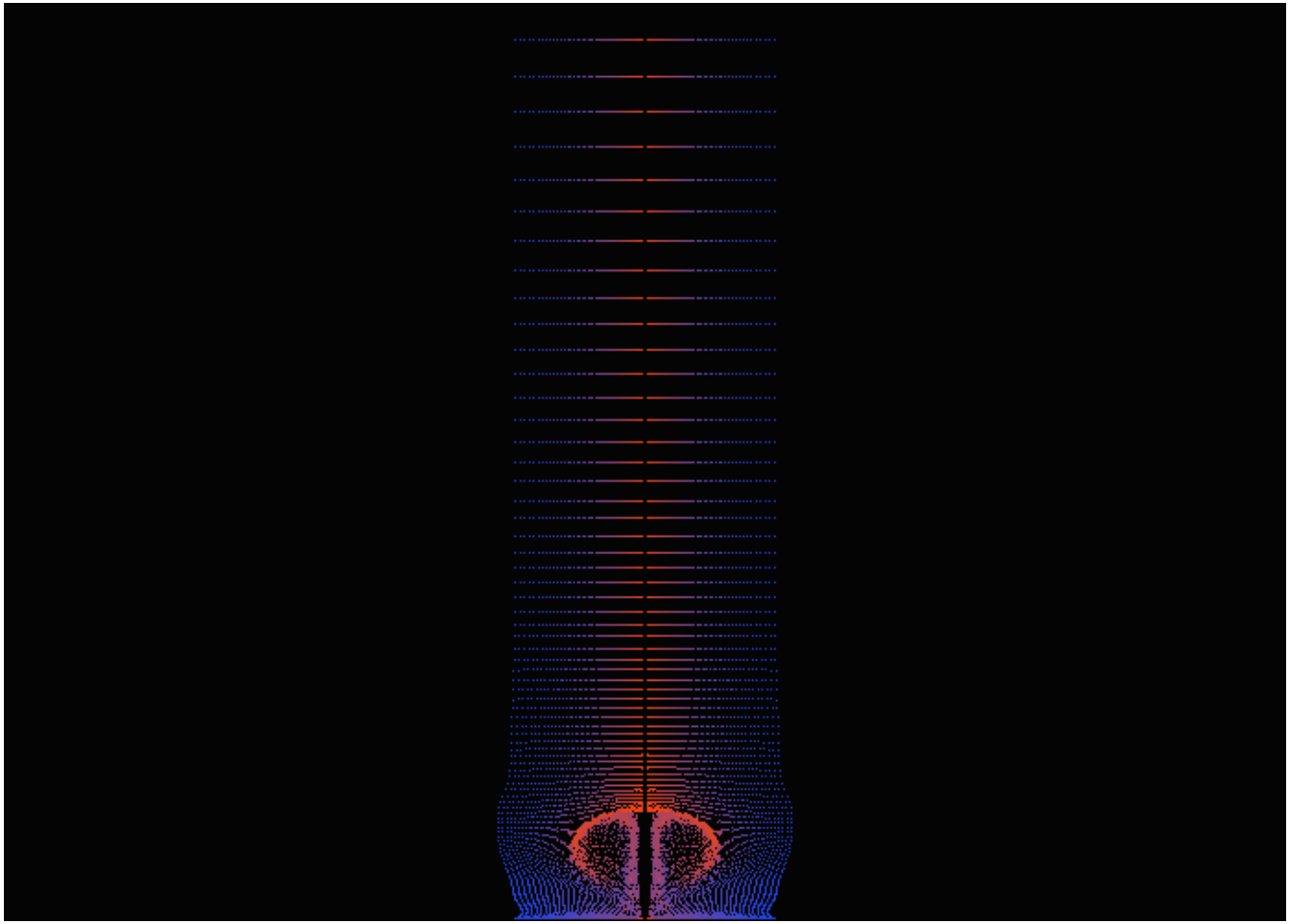


сетка 1000x1000



Два кластера
33+12
процессоров

Траектории пробных частиц



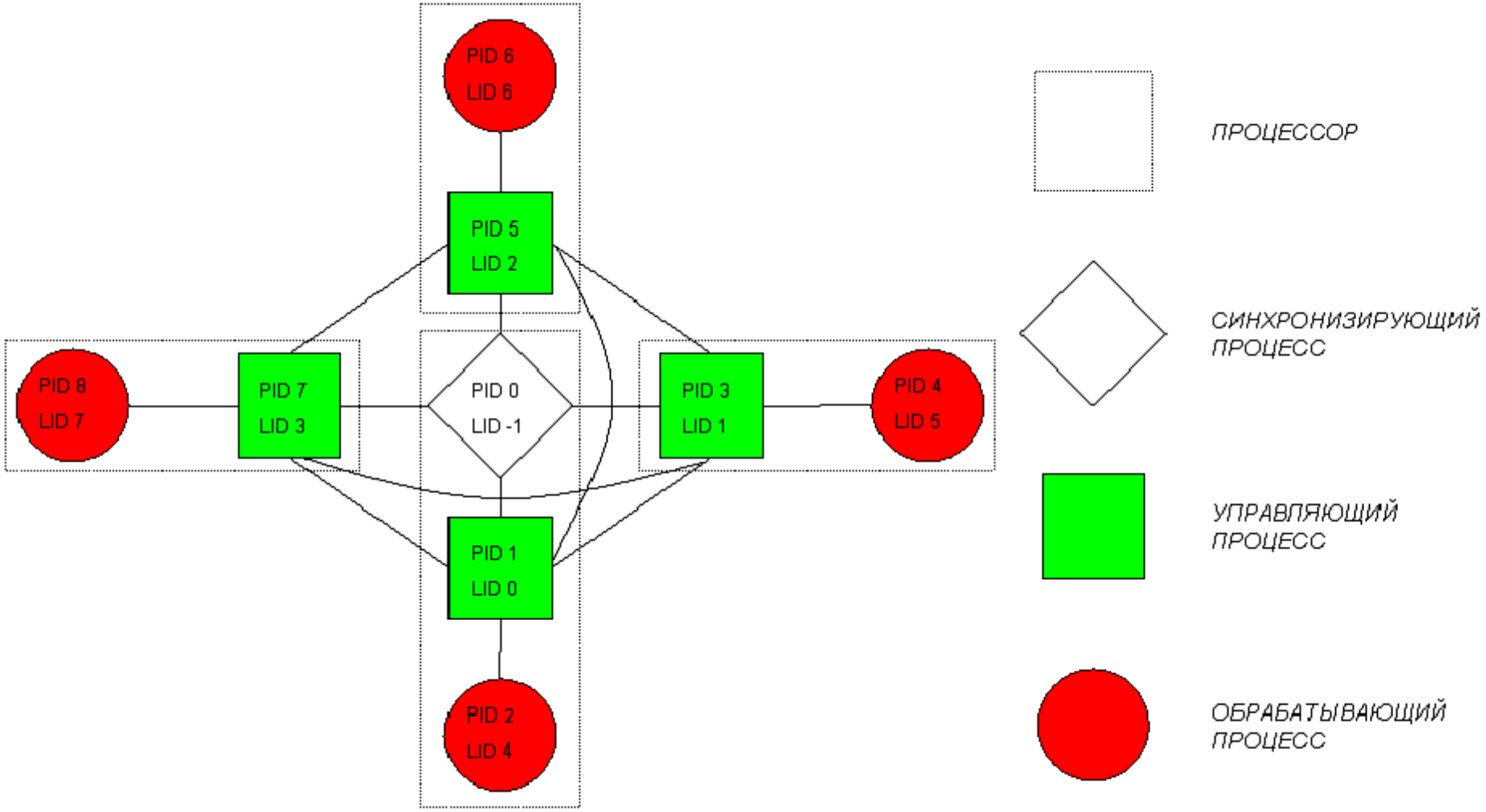
статическая балансировка

Число процессоров	Время	Ускорение
1	289	1.00
2	141	2.04
3	142	2.02
4	147	1.96
5	283	1.02
6	147	1.96
7	238	1.21
8	161	1.79

Динамическая балансировка загрузки

Число физических процессоров	Число процессов	Время	Ускорение	Эффективность
5	5(2)	158.18	1.8284865	91%
7	7(3)	107.5	2.6905116	90%
9	9(4)	82.95	3.4867993	87%
2	5(2)	113.85	2.540448	127%
3	7(3)	85.75	3.3729446	112%
4	9(4)	70.33	4.1124698	103%
5	11(5)	60.61	4.7719848	95%
6	13(6)	52.56	5.5028539	92%
7	15(7)	43.88	6.5913856	94%
8	17(8)	39.7	7.2853904	91%

Схема взаимодействия процессов



Типы процессов

- Тип процесса может быть определен с помощью функции **proc_type(void)**
- Синхронизирующий процесс T_MASTER обеспечивает согласованность выполнения расчета и записи результатов
- Управляющие процессы T_TASK отвечают за хранение данных, выполнение «диффузионного блока» и распределение заданий на этапе динамической балансировки загрузки.
- Обрабатывающие процессы T_CHEM выполняют обработку заданий из списков, формируемых на этапе динамической балансировки.

```
#include < my_net_mpi.h>
int main(int argc, char *argv[])
{
MPI_Init(&argc, &argv);
…
        init_net(argc, argv);
        logik_init();
…
        if(proc_type()==T_MASTER)       main_master( argc, argv);
        if(proc_type()==T_TASK)   main_task();
        if(proc_type()==T_CHEM)  ChemThread();
…
MPI_Finalize();

}
```

```
void main_master(int argc, char *argv[])
{
for(j=0;j<ITT;j++)
        {
        BarrierFarm();
        …
        BarrierFarm();
        }
}
```

```
for(I=0;I<ITT;I++)
        {
        // выполнение диффузионного блока
        GD();
        // обработка легковесных узлов
                {
                for(ii=0; ii<n1*n2; ii++)
                        {
                        i = ii / n2;  j = ii % n2;

                        if(!TestForCalculating(ii))
                                {
                                double y[2] = {fn[i][j], fm[i][j]}, r0, r1;

                                RungeStep(y[0], y[1], tau, &r0, &r1);
                                N_PLUS[i][j] = r0; M_PLUS[i][j] = r1;
                                }
                        } // for ii
                }
```

```
                                        RungeStep(y[0], y[1], tau, &r0, &r1);
                                        N_PLUS[i][j] = r0; M_PLUS[i][j] = r1;
                                        }
                                } // for ii
                        }
// обработка узлов с помощью динамической балансировки
                BarrierFarm();
                MainServer();
                BarrierFarm();

                        {// ПЕРЕХОД НА НОВЫЙ СЛОЙ ПО ВРЕМЕНИ
                        double **TMP;
                        TMP = N; N = N_PLUS; N_PLUS = TMP;
                        TMP = M; M = M_PLUS; M_PLUS = TMP;
                        }

                t += tau;
                }

FinishChemThread();
}
```

ChemThread – ***процедура обрабатывающего процесса***

Эта процедура не подлежит модификации пользователем. Вызов библиотечной процедуры *ChemThread* полностью обеспечивает выполнение всех действия необходимых для вызова обрабатывающей функции ***ch4*** для всех заданий, обрабатываемых с помощью динамической балансировки.

PrepareOwnPointList - ***формирование списка заданий***

Каждый управляющий процесс на основе имеющихся в его памяти данных формирует список заданий. Каждое задание описывается структурой **USER_TASKPOINT**. Определение данного типа дается в заголовочном файле user_task.h. Для формирования списка заданий управляющим процессом вызывается функция **int PrepareOwnPointList(void)** (функция определяется в файле user_task.c). Внутри функции необходимо определить переменную **n_of_our_points**, в которую записывается число обрабатываемых в ходе выполнения итерации заданий. Далее при формировании непосредственно списка заданий **n_of_our_points** раз будет вызвана функция **int ExtractChemData(int name, USER_TASKPOINT *buf)** (функция определяется в файле main1.c), где **name** – порядковый номер узла (задания), а **buf** – указатель на структуру, в которую будут записаны данные для обработки узла.

ExtractChemData ***– запись данных одного задания в список заданий***

Подпрограмма заносит все данные, необходимые для автономной обработки одного задания в список заданий.

```
int ExtractChemData(int name, USER_TASKPOINT *buf)
{
int i, j;

i = name / n;
j = name % m;

buf->f[0] = _F_[i][j];
buf->f[1] = _F_[i-1][j];
buf->f[2] = _F_[i+1][j];
buf->f[3] = _F_[i][j-1];
buf->f[4] = _F_[i][j+1];

return 0;
}
```

StoreChemData ***- запись результатов обработки***

StoreChemData является обратной по отношению к процедуре *ExtractChemData*

```
int StoreChemData(int name, USER_TASKPOINT *p)
{
        int i, j;

        i = name / n;
        j = name % m;

        _F_[i][j] = p->f[0];

        return 0;
}
```

ch4 ***- обработка элементарных заданий***

Процесс динамического перераспределения загрузки процессоров скрыт от пользователя. Задания автоматически перераспределяются и передаются обрабатывающим процессам, которые осуществляют для каждого полученного задания вызов функции **int ch4(USER_TASKPOINT *ptask)** (функция определяется в файле main1.c) и передают результат работы обратно управляющему процессору. Результат работы записывается в одно из полей структуры **USER_TASKPOINT**.

```
int ch4(USER_TASKPOINT *ptask)
{
ptask->f[0] = FFF(ptask->f[0], ptask->f[1], ptask->f[2], ptask->f[3], ptask->f[4]);
return 1;
}
```

TestForCalculating ***–определение предполагаемого времени обработки узла***

Проверка необходимости включения узла в список выполняется с помощью функции **int TestForCalculating(int name)** (функция определяется в файле main1.c), которая возвращает 1 при необходимости включения задания в список динамической балансировки и 0 – в противном случае.

```
int TestForCalculating(int name)
{
        int i, j;
        i = name / n;
        j = name % m;

        if(_F_[i][j] == 0.0)        return 0;
        else                        return 1;
}
```

1. **Алгоритм распределенной обработки данных для балансировки загрузки при решении задач химической кинетики**
2. Критерий «горячих» точек
3. Обработка своих точек - в первую очередь
4. Отсутствие
   - централизованного сбора данных
   - централизованного управления
5. Наличие
   - «рынка» для перераспределения загрузки:
     - занятые процессоры
     - свободные процессоры
     - дисциплина взаимодействия процессоров
6. Минимизация коммуникаций во время счета
7. Одновременность счета и коммуникаций

Серверный параллелизм

- Серверный параллелизм - метод динамической балансировки загрузки, применимый в случае обработки ***распределенного*** множества элементарных заданий ***непредсказуемой*** трудоёмкости

Литература

- Якобовский М.В. Введение в параллельные методы решения задач: Учебное пособие / Предисл.: В. А. Садовничий. – М.: Издательство Московского университета, 2013. – 328 с., илл. – (Серия «Суперкомпьютерное образование») ISBN 978-5-211-06382-2
- М.В.Якобовский, С.А.Суков, Динамическая балансировка загрузки, 2004 http://lira.imamod.ru/mipt201403/CourseManualBalance.pdf
- М.В.Якобовский, С.А.Суков, Библиотека подпрограмм динамической балансировки загрузки, 2004 http://lira.imamod.ru/mipt201403/DynamicLibrary.pdf

Контакты

Якобовский М.В.

проф., д.ф.-м.н.,

зав. сектором

«Программного обеспечения многопроцессорных систем и вычислительных сетей»

Института прикладной математики им. М.В.Келдыша Российской академии наук

mail: lira@imamod.ru

web: http://lira.imamod.ru