Разработване на паралелни програмни приложения

част 2

Паралелни алгоритми за сортиране

- Използват разпределен входен списък на сортираните стойности

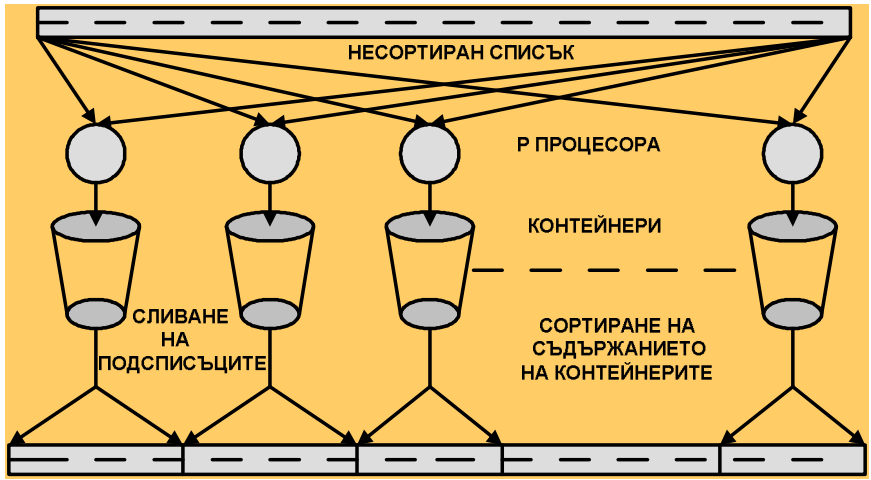
- Резултатът също е разпределен между процесите
  - налага се да се изпълни операция по “събирането” му (merge)
  - след сортирането отделните процесори съхраняват сортирани части на изходния списък с елементи

Паралелно сортиране с контейнери

- *Задача*
  - да се сортират числа с алгоритъм с контейнери (*bucket sort*)
- *Целева паралелна компютърна платформа*
  - клъстер от работни станции
- *Програмен модел*
  - плосък (flat)
- *Имплементация*
  - MPI

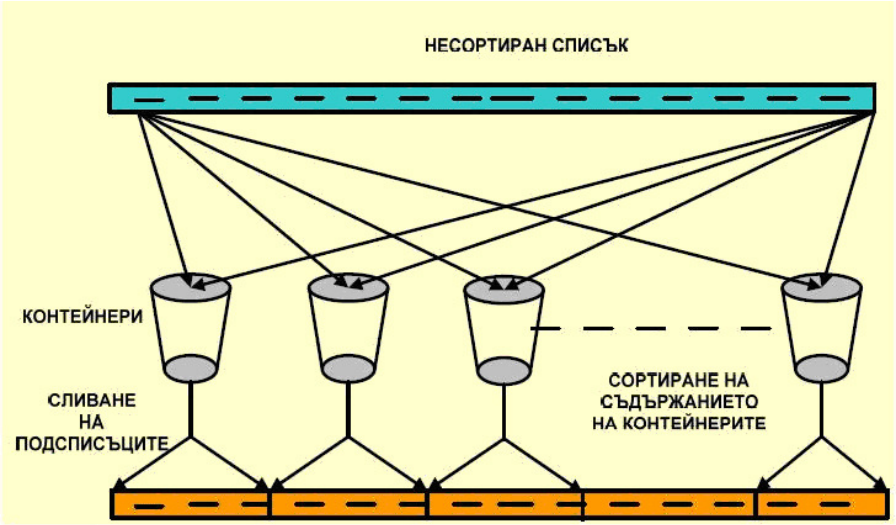
Паралелно сортиране с контейнери

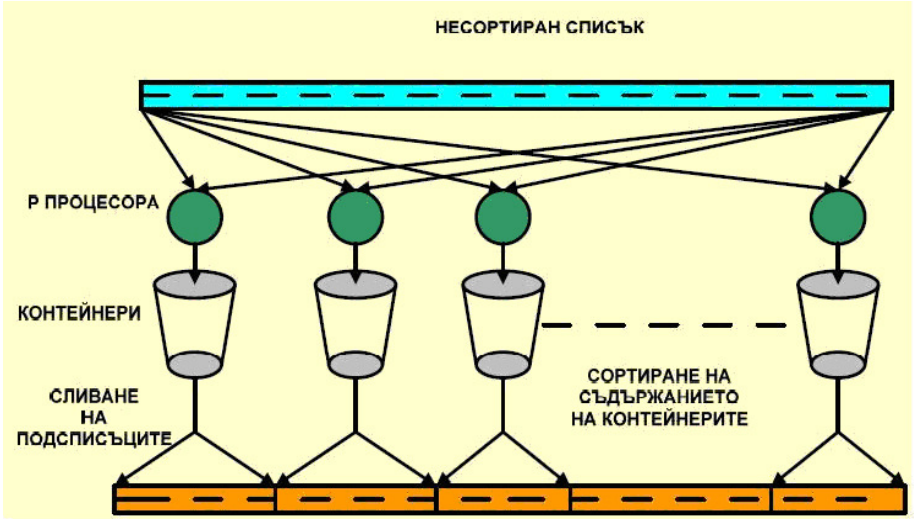
- Сортиране с контейнери
  - алгоритъм за сортиране чрез разделяне на интервала от стойности на масива на краен брой контейнери
  - всеки контейнер съдържа стойностите в определен числов интервал
- Алгоритъмът е ефективен при равномерно разпределение на стойностите на елементите в даден интервал
  - например от 0 до a–1



Паралелно сортиране с контейнери

- Интервалът се разделя на *m* равни области
  - от 0 до *a*/*m*–1, от *a*/*m* до 2*a*/*m*–1 и т.н.
  - всеки контейнер съдържа елементи със стойности, принадлежащи на интервалите на разделяне
  - числата в контейнерите се сортират с използване на последователен алгоритъм за сортиране





Паралелно сортиране с контейнери

- Алгоритмични парадигми
  - „главен – подчинен” и SPMD

- Главният процес е отговорен за следните задачи
  - изпраща на подчинените процеси интервалите на локалните контейнери (*MPI_Send()*)
  - изпраща на подчинените процеси елементите за сортиране в локалните контейнери (*MPI_Send()*)
  - получава от подчинените процеси частичните резултати (*MPI_Recv()*)
  - обединява локално сортираните от подчинените процеси данни и формира изходния сортиран масив

Паралелно сортиране с контейнери

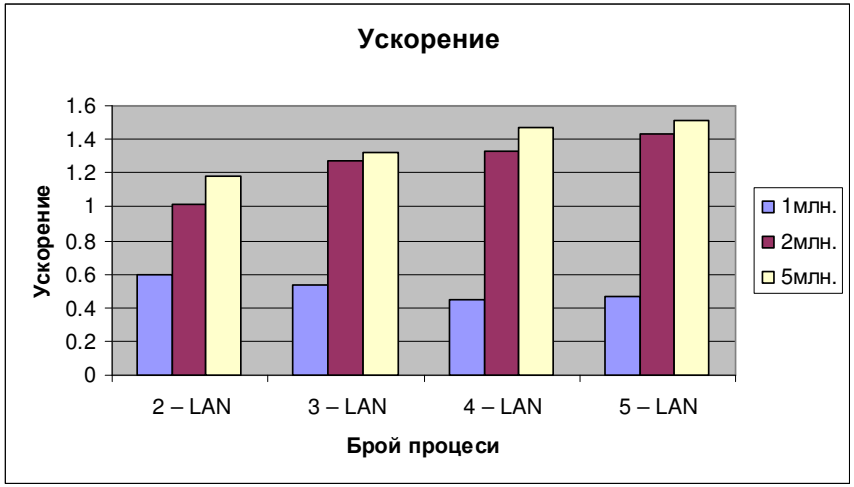
- Всеки подчинен процес използва локален контейнер
  - елементите в локалните контейнери се сортират паралелно
- Подчинените процеси изпълняват следните дейности
  - получават от главния процес интервалите и размера на локалните контейнери (*MPI_Recv()*)
  - получават от главния процес стойностите за сортиране (*MPI_Recv()*)
  - извършват разпределяне на елементите в локалните контейнери съгласно интервалите на стойностите
  - изпращат на главния процес локално получените частични резултати (*MPI_Send()*)

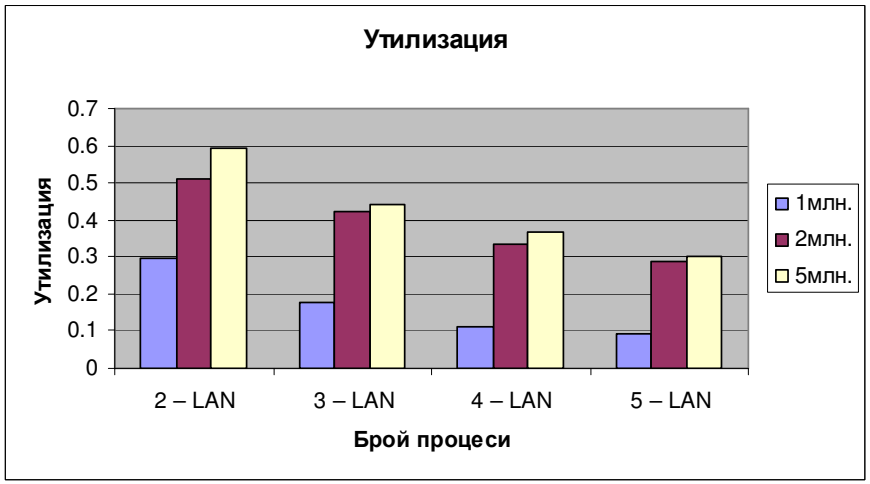
Паралелно сортиране с контейнери

- Модел на комуникация между процесите

Паралелно сортиране с контейнери

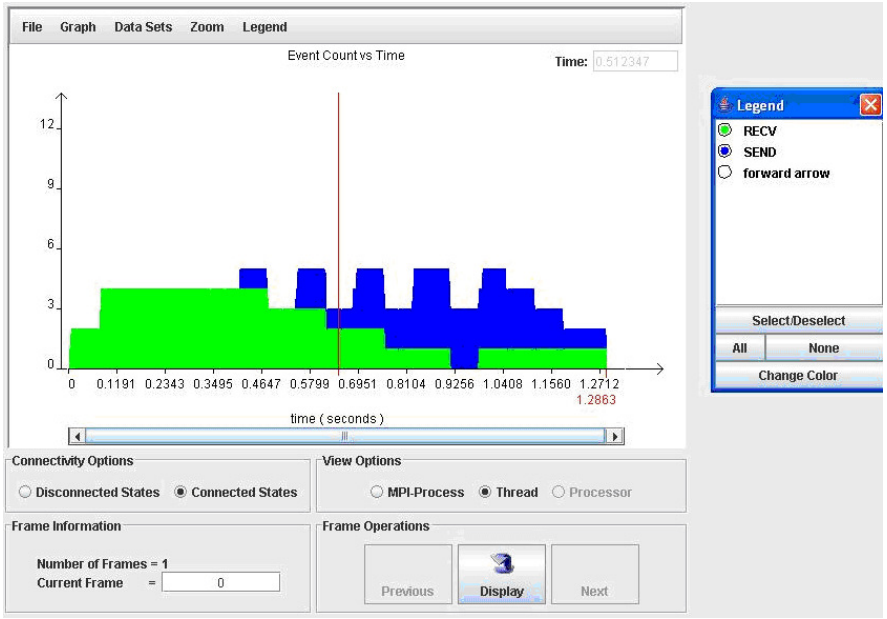
- Производителност
  - ускорение
  - утилизация

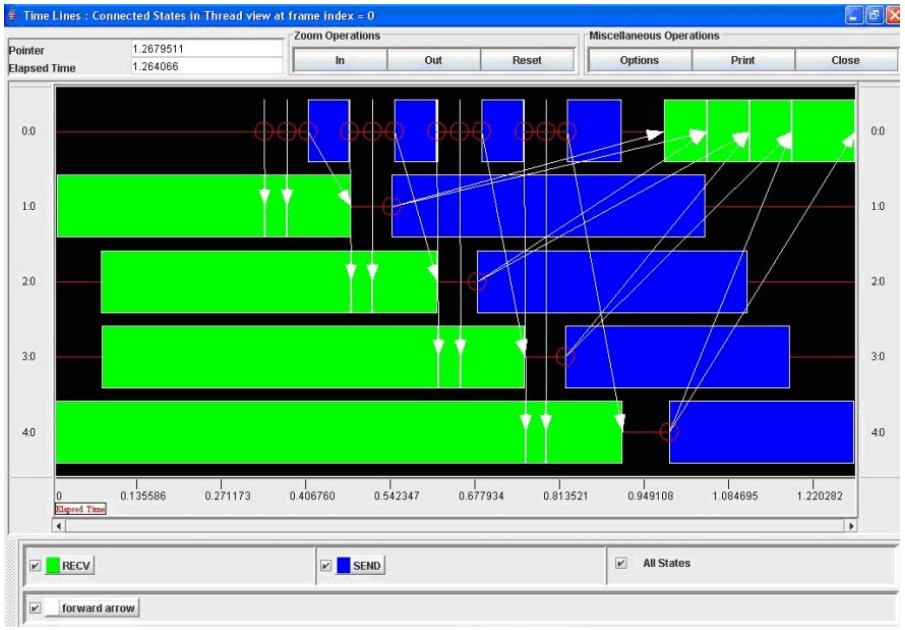




Паралелно сортиране с контейнери

- Профили на паралелизмите
- Диаграма на Гант





Паралелно пирамидално сортиране

- *Задача*
  - да се сортират числа с алгоритъм за пирамидално сортиране (*heap sort*)
- *Целева паралелна компютърна платформа*
  - клъстер от многоядрени възли
- *Програмен модел*
  - Хибриден (обмен на съобщения + многонишкова обработка)
- *Имплементация*
  - MPI + OpenMP

Паралелно пирамидално сортиране

- Алгоритъм за пирамидално сортиране
  - добра ефективност при най-лоша конфигурация на входните данни
  - по-бавен в сравнение с други алгоритми за сортиране с разделяне
  - предимство
    - не изисква допълнителна памет за съхраняване на междинни резултати
    - работи директно с входното множество от данни
    - подходящ за сортиране на големи по обем данни
  - базира се на разполагане на елементите на сортирания масив в балансирано двоично дърво – пирамида, с височина *h*

Паралелно пирамидално сортиране

- За да се намали размера на заеманата памет пирамидата може ефективно да се представи с *n* мерен масив като наследствените връзки могат да се опишат с проста линейна функция на позициите им
  - например наследниците на *i*-тия възел ще бъдат 2*i* и 2*i*+1
  - тогава $h_i \geq h_{2i}$ и $h_i \geq h_{2i+1}$
  - в този случай пирамидата може да се помести във все още несортираната част от входния масив

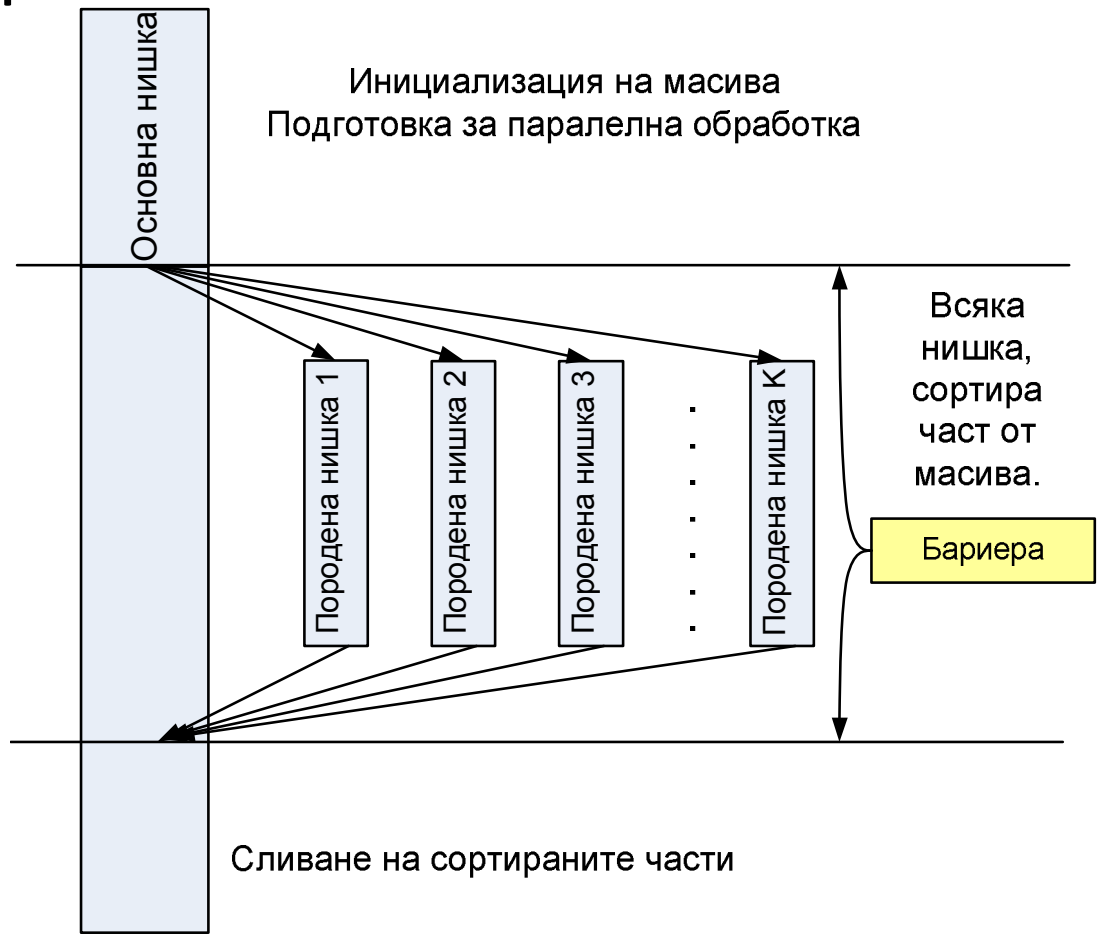
- Алгоритъмът сортира данните като последователно премахва най-голямата стойност и я поставя в края на сортирания масив докато се изчерпят елементите в пирамидата и се попълни сортираният масив

Паралелно пирамидално сортиране

- Зависимост между итерациите на алгоритъма
- Паралелизиране с многонишкова обработка чрез декомпозиция по данни
  - всяка нишка сортира част входния масив
  - след което сортираните части се сливат
- Този подход е неефективен за малки по размер масиви за сортиране, тъй като времето за управление на нишките и за сливане на частичните резултати става съизмеримо с времето по действителното сортиране

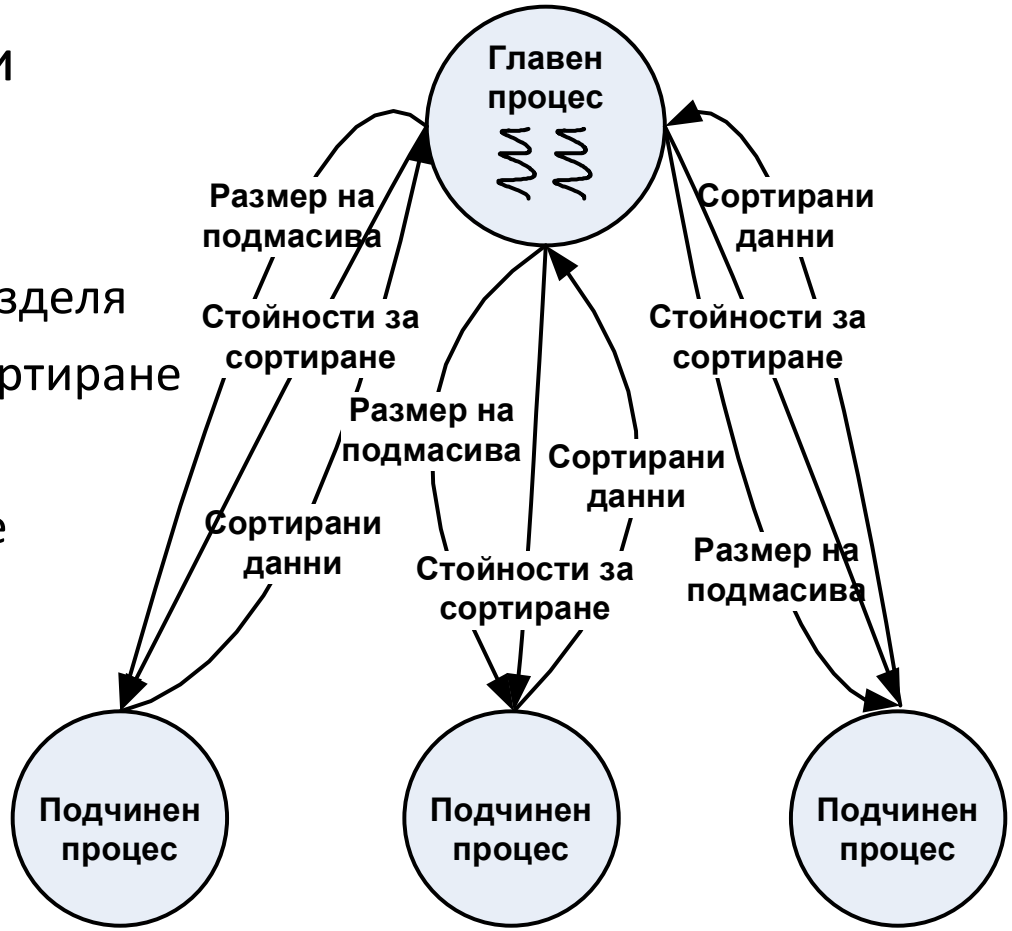
Паралелно пирамидално сортиране

- Модел на паралелните изчисления



Паралелно пирамидално сортиране

- Модел на комуникациите между процесите
- Алгоритмични парадигми
  - „главен - подчинен” и декомпозиция на данните
  - несортирания масив се разделя на части, изпраща се за сортиране на подчинените процеси, които връщат сортираните части на главния процеса за да извърши сливането



Паралелно пирамидално сортиране

- Главният процес извършва следните дейности
  - изпраща на всеки подчинен процес размера на частта от масива за локално сортиране (*MPI_Send()*)
  - изпраща на всеки подчинен процес елементите от масива, които той трябва да се сортират локално (*MPI_Send()*)
  - сортира своята част от масива, като изпълнява програмния код на подчинен процес с използване на многонишкова обработка за ускоряване на изчисленията
  - получава от подчинените процеси локално сортираните части от масива (*MPI_Recv()*)
  - слива сортираните части за получаване на сортирания изходен масив като използва многонишкова обработка за ускоряване на изчисленията

Паралелно пирамидално сортиране

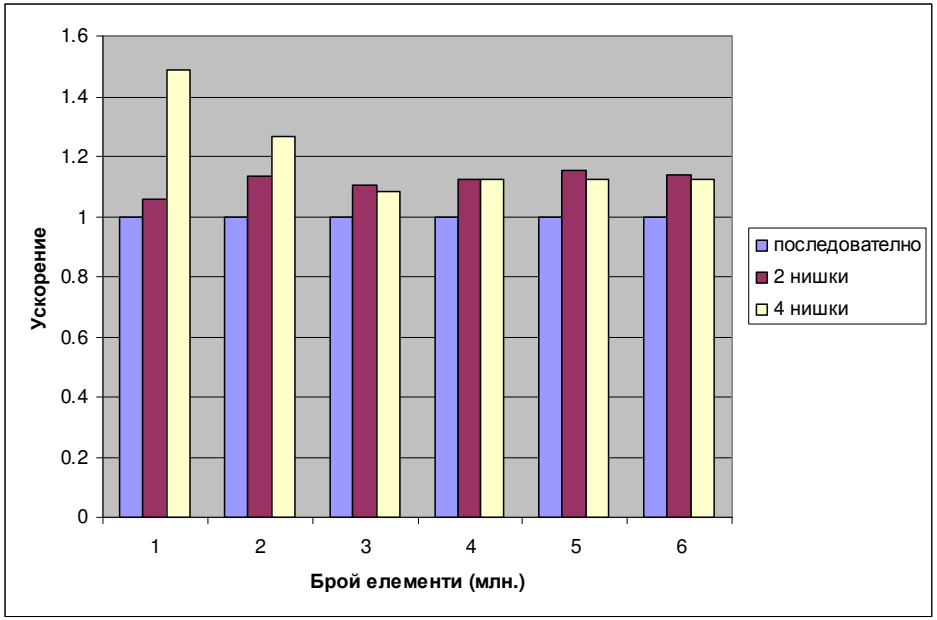
- Всеки от подчинените процеси изпълнява следните задачи
  - получава от главния процеса размера на частта от масива, която ще сортира (*MPI_Recv()*)
  - получава от главния процес елементите за сортиране (*MPI_Recv()*)
  - сортира своята част от масива с използване на алгоритъм за пирамидално сортиране
  - изпраща на главния процеса сортираната последователност от стойности (*MPI_Send()*)

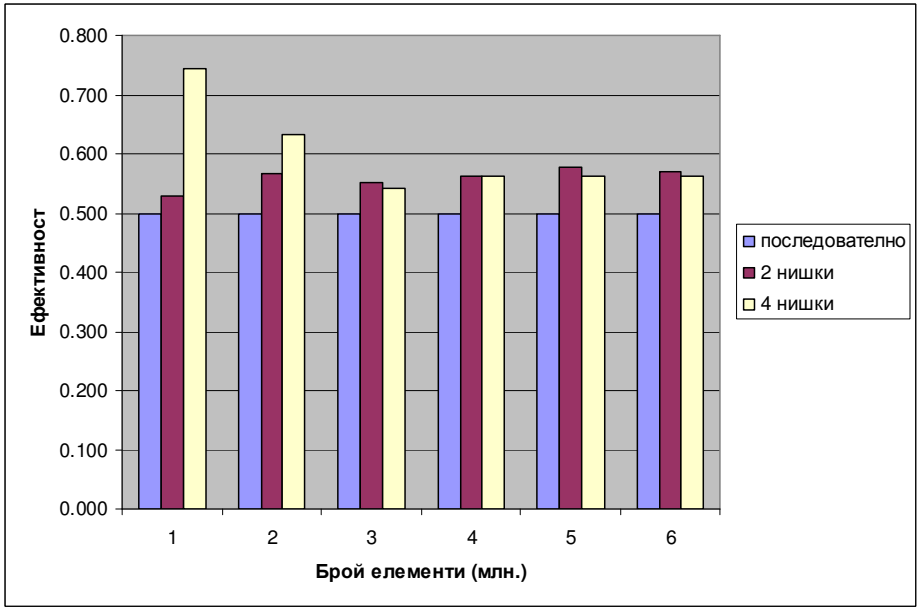
Паралелно пирамидално сортиране

- Всеки от подчинените процеси изпълнява следните задачи
  - получава от главния процеса размера на частта от масива, която ще сортира (*MPI_Recv()*)
  - получава от главния процес елементите за сортиране (*MPI_Recv()*)
  - сортира своята част от масива с използване на алгоритъм за пирамидално сортиране
  - изпраща на главния процеса сортираната последователност от стойности (*MPI_Send()*)

Паралелно пирамидално сортиране

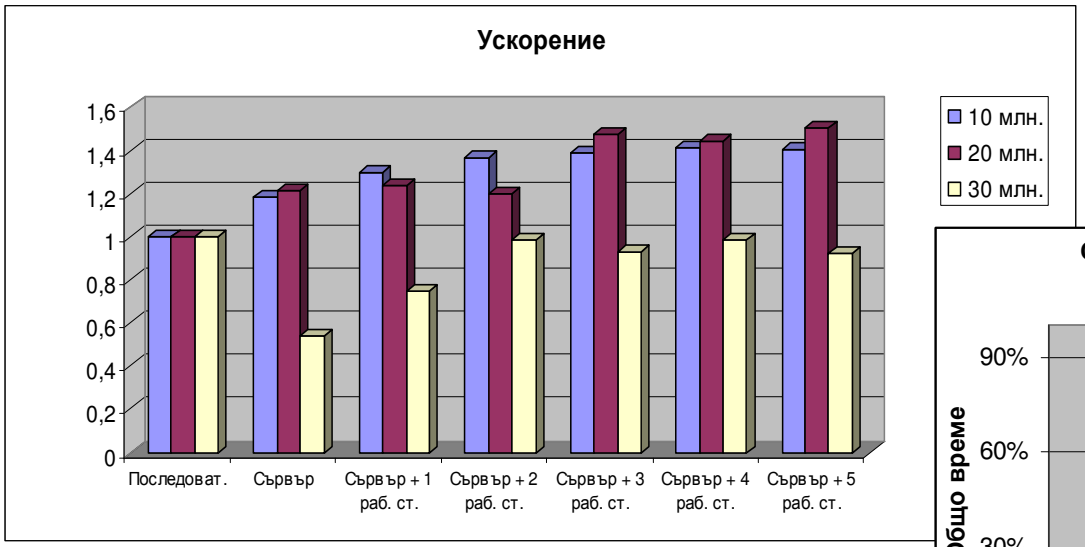
- Производителност на многонишков модел
  - ускорение
  - утилизация

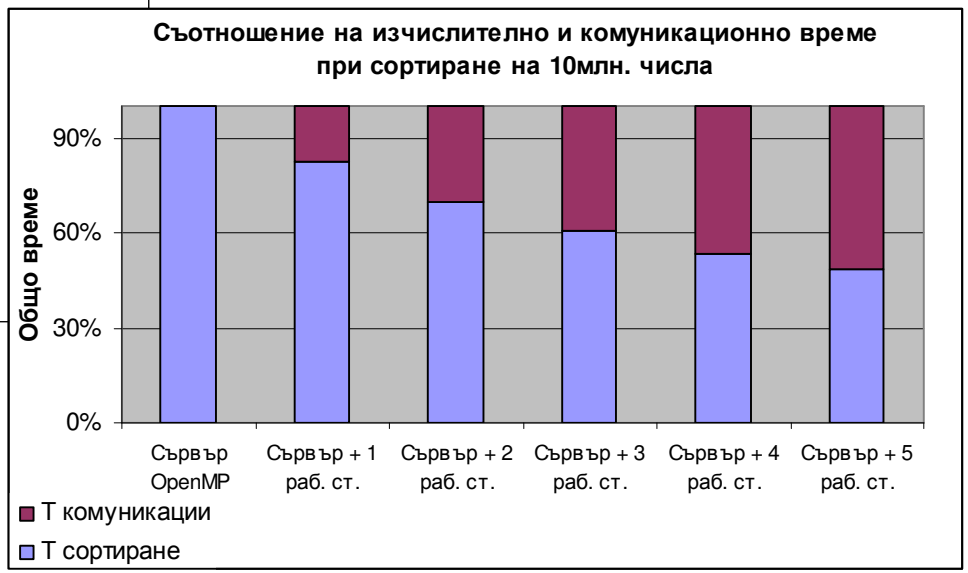




Паралелно пирамидално сортиране

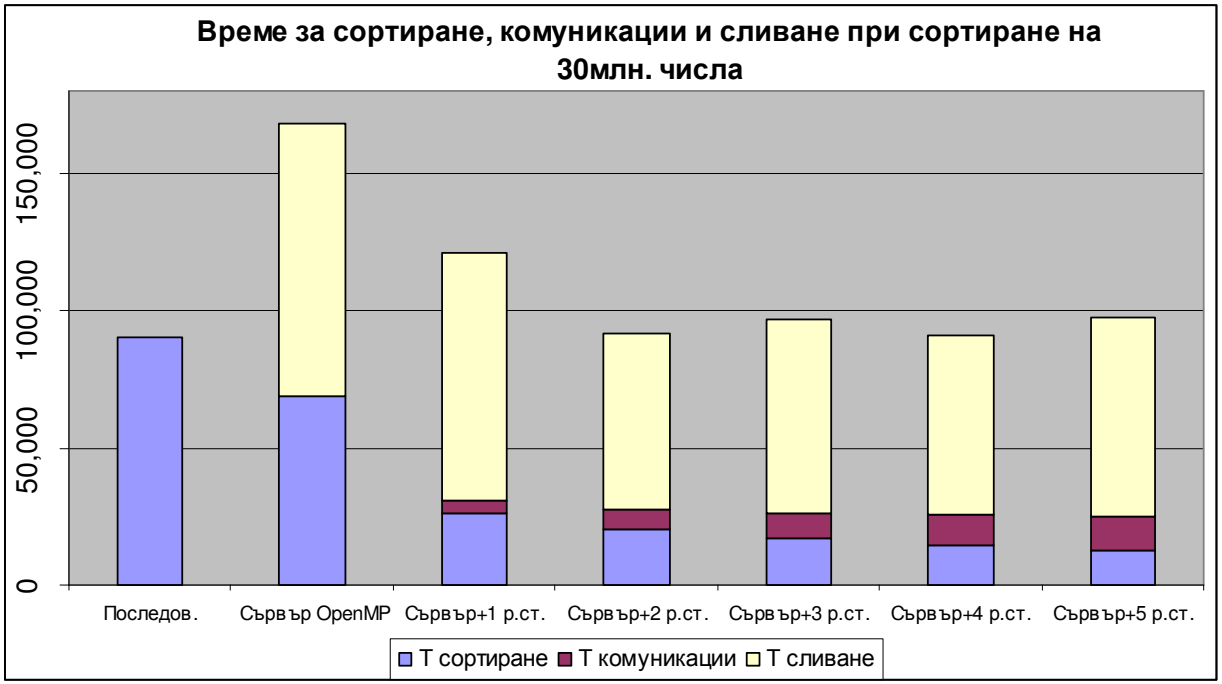
- Производителност на хибриден модел
  - ускорение
  - съотношение време за сортиране – време за комуникация





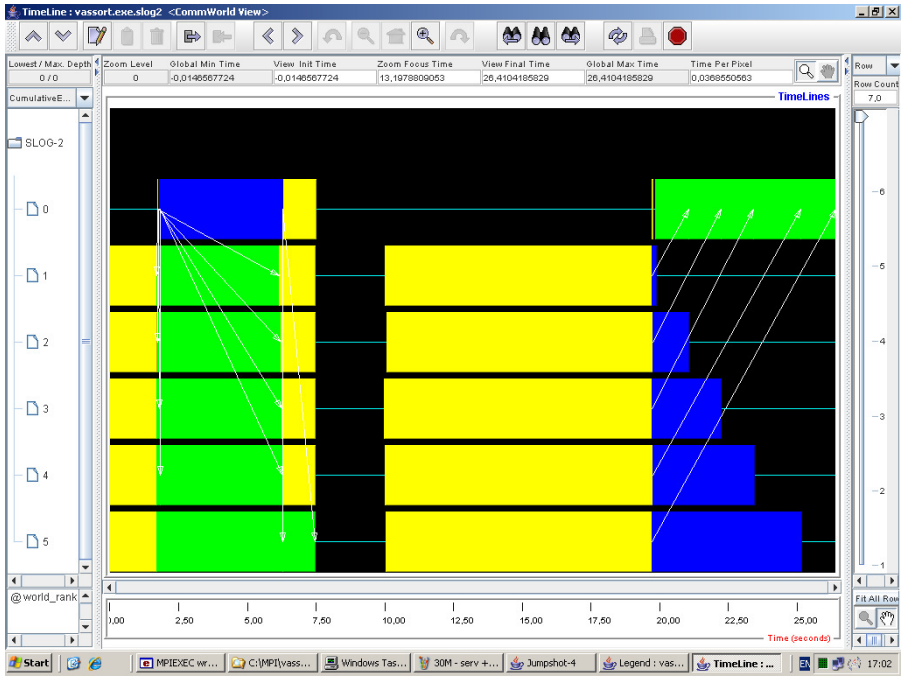
Паралелно пирамидално сортиране

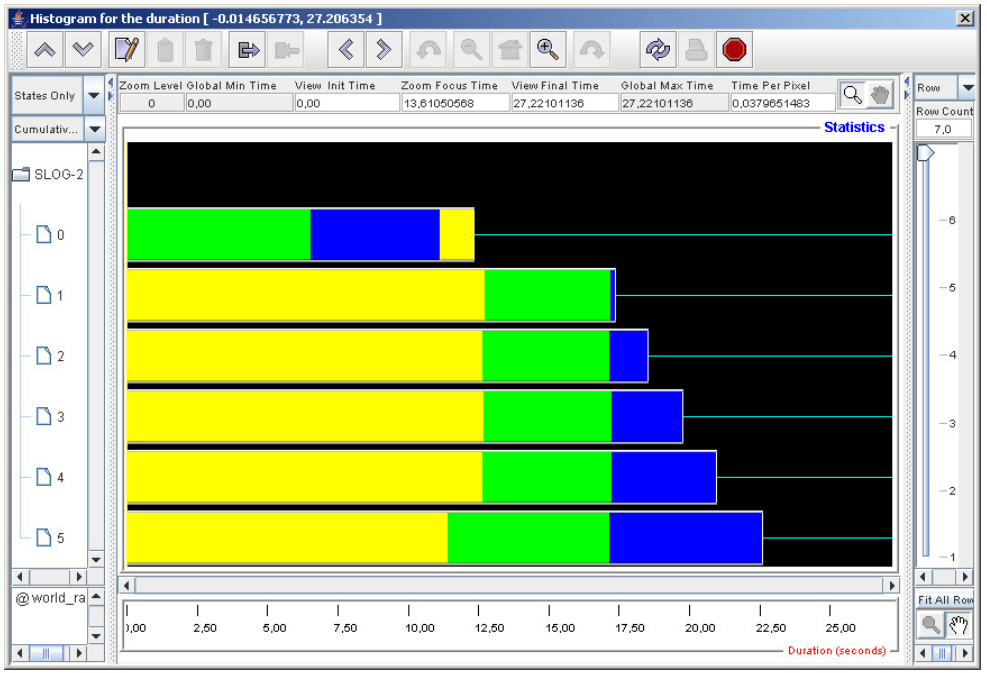
- Производителност на хибриден модел
  - съотношение време за сортиране – време за комуникация – време за сливане



Паралелно пирамидално сортиране

- Производителност на хибриден модел
  - диаграма на Гант и профил на комуникациите



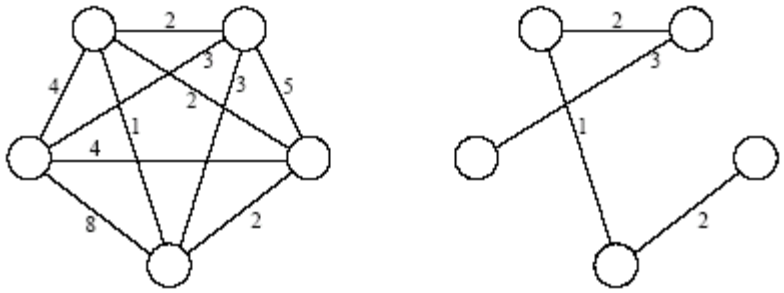


Минимално скелетно дърво

- *Задача*
  - да се намери минимално скелетно дърво в свързан теглови граф с алгоритъм на Прим
- *Целева паралелна компютърна платформа*
  - клъстер от работни станции
- *Програмен модел*
  - плосък (flat)
- *Имплементация*
  - MPI

Минимално скелетно дърво

- Скелетно дърво на неориентиран граф
  - подграф, който е дърво и съдържа всички възли на графа
- Тегло на подграф в теглови граф
  - сумата на всички тегла на дъги в подграфа
- Минимално скелетно дърво (*minimum spanning tree* (MST) за теглови неориентиран граф
  - скелетно дърво с минимално тегло



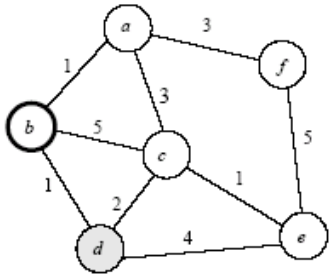
Минимално скелетно дърво

- Алгоритъмът на Прим
  - построява минимално скелетно дърво на граф с n върха за n–1 итерации
  - в началото се избира произволен възел, който се добавя към текущото минимално скелетно дърво
  - текущото минимално скелетно дърво се увеличава с добавяне на връх, който е свързан с най-малко тегло с вече включените в дървото възли
    - от необходените възли се избира този, който е свързан с ребро с минимално тегло с вече обходен възел

Минимално скелетно дърво

- Алгоритъмът на Прим

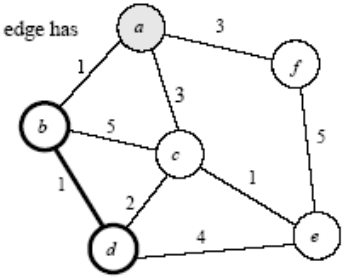
(a) Original graph



	a	b	c	d	e	f
d[]	1	0	5	1	∞	∞

	a	b	c	d	e	f
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

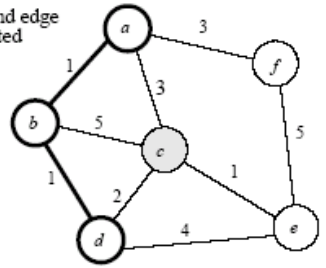
(b) After the first edge has been selected



	a	b	c	d	e	f
d[]	1	0	2	1	4	∞

	a	b	c	d	e	f
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

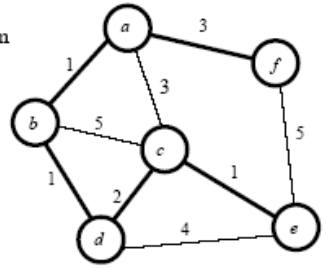
(c) After the second edge has been selected



	a	b	c	d	e	f
d[]	1	0	2	1	4	3

	a	b	c	d	e	f
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

(d) Final minimum spanning tree



	a	b	c	d	e	f
d[]	1	0	2	1	1	3

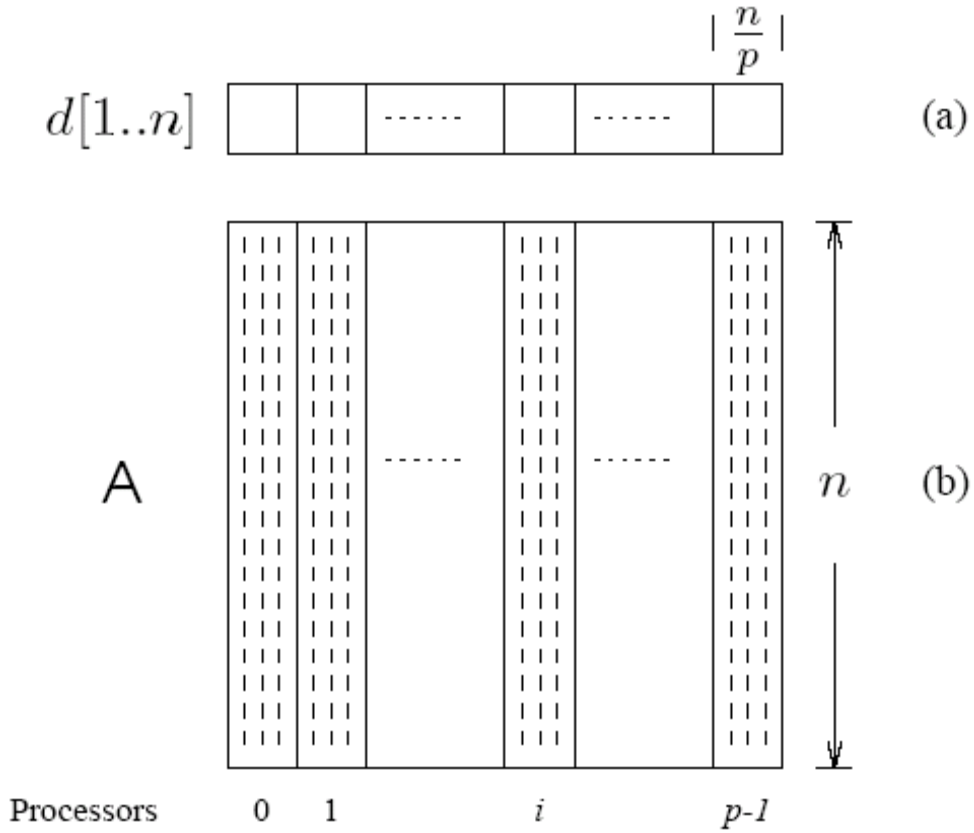
	a	b	c	d	e	f
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

Минимално скелетно дърво

- Паралелен алгоритъм на Прим
  - алгоритъмът се изпълнява в n-1 итерации, при които има зависимост на изчисленията
    - не могат да се изпълняват паралелно
  - паралелни изчисления се прилагат за определяне на следващият възел за добавяне към минималното скелетно дърво на всяка итерация на алгоритъма
- Използва се декомпозиция по данни
  - множеството възли, от които се избира възел за добавяне към текущото дърво, се разпределя между процесите
  - матрицата на съседство на графа се разделя на едномерни блокове
  - векторът за разстояние също се разделя между процесите

Минимално скелетно дърво

- Декомпозиция по данни при паралелен алгоритъм на Прим



Минимално скелетно дърво

- Паралелен алгоритъм на Прим
  - на всяка стъпка всеки процес определя локално възел за добавяне към минималното скелетно дърво
  - чрез глобална комуникация се определя глобално възелът с най-малко тегло на дъга възел от дървото
  - избраният за добавяне възел се изпраща на всички процеси
  - процесите обновяват локално съхраняваното минимално скелетно дърво

Минимално скелетно дърво

- Процесът-мениджър е отговорен за извършване на следните дейности
  - Изпраща на всички процеси матрицата на съседство за обработвания граф с функцията *MPI_Bcast()*
  - Изпраща на всички процеси индексът на връх, който последно е добавен към минималното скелетно дърво с функцията *MPI_Bcast()*
  - Събира от подчинените процеси изчислените стойности за частите от масива с минимални тегла до възел от минималното скелетно дърво с функцията *MPI_Scatter()*
  - Избира връх за добавяне към минималното скелетно дърво на базата на резултатите, получени от подчинените процеси

Минимално скелетно дърво

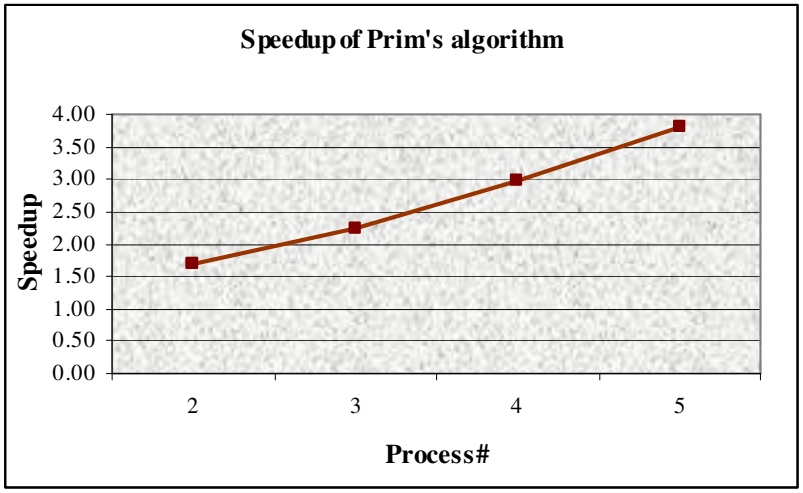
- Процесите-работници имат следните задачи
  - Получават от процеса-мениджър матрицата на съседство за обработвания граф с функция *MPI_Bcast()*
  - Получават от процеса-мениджър връх, който е текущо добавен към минималното скелетно дърво с функция *MPI_Bcast()*
  - Определят за всеки от върховете в собственото им подмножество ребро с най-малка тежест
  - Изпращат на процеса-мениджър получените стойности с функцията *MPI_Scatter()*

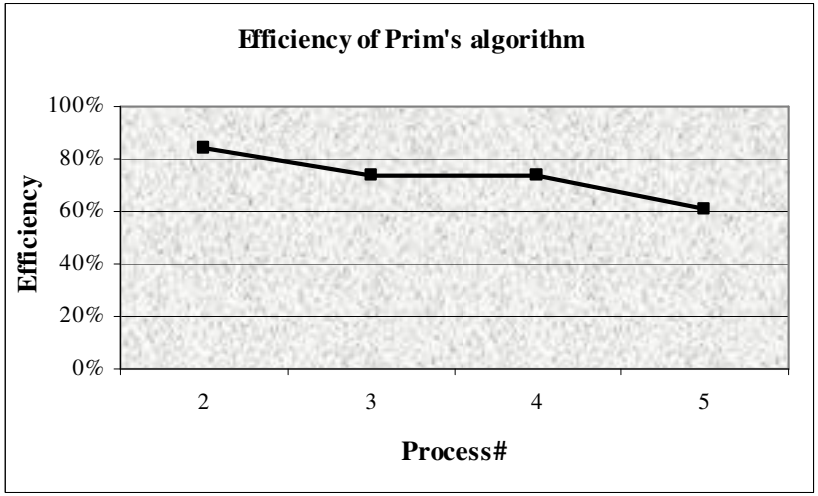
Минимално скелетно дърво

- Модел на комуникация между процесите

Минимално скелетно дърво

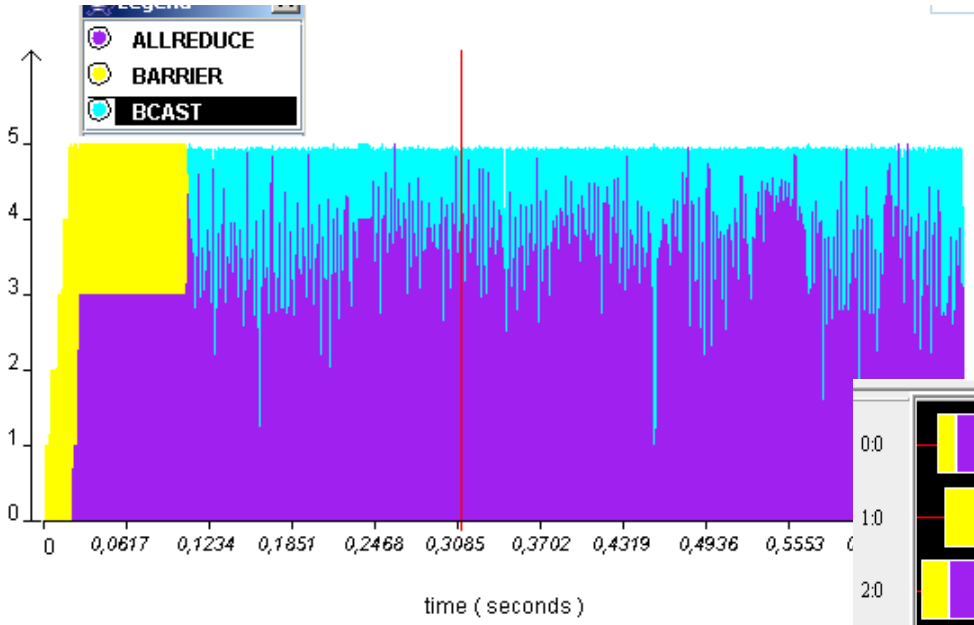
- Производителност
  - почти линейно ускорение
  - висока ефективност на използване на процесорите
  - добро мащабиране

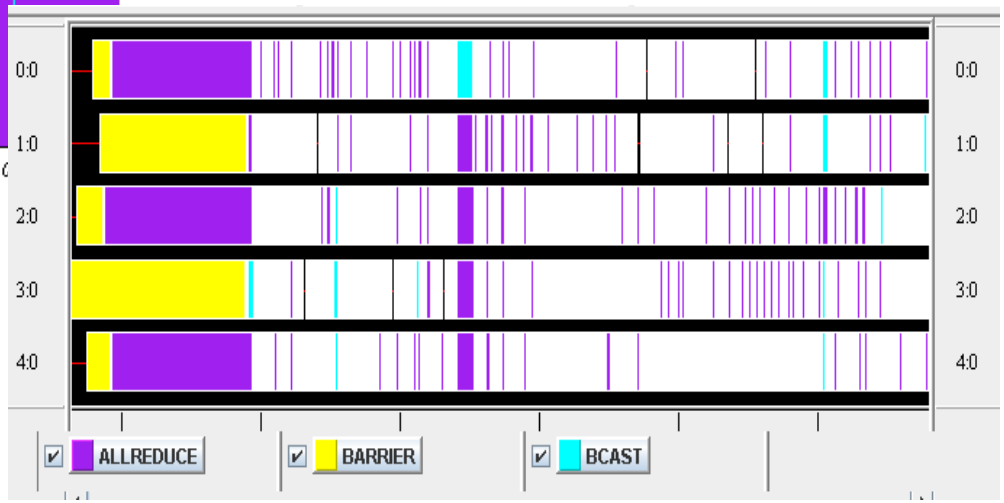




Минимално скелетно дърво

- Профил на комуникациите и диаграма на Гант





Най-кратък път в граф

- Намиране на най-кратък път в граф от един възел до всички останали
  - Single-Source Shortest Paths
- Намиране на най-кратък път в граф между всеки два възела
  - All-Pairs Shortest Paths

Най-кратък път в граф

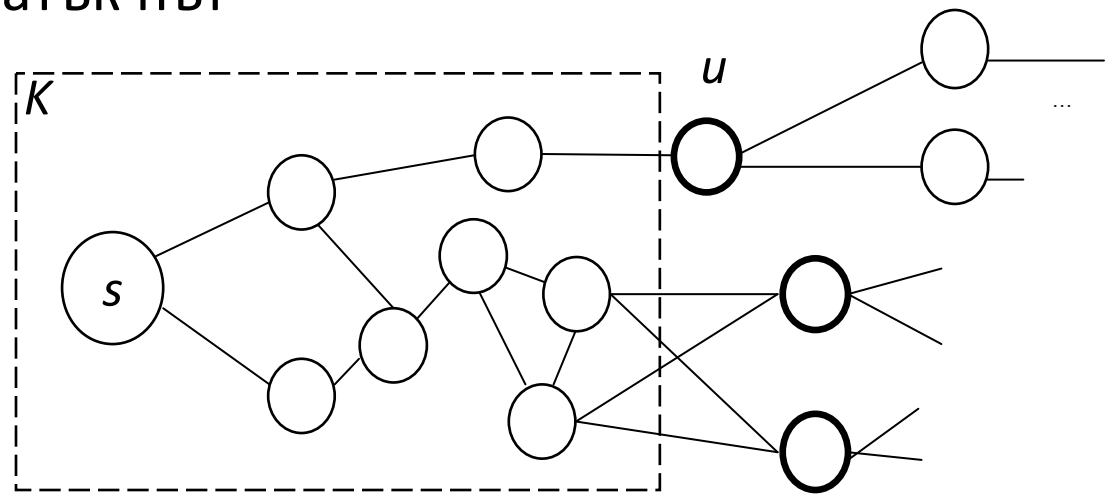
- *Задача*
  - да се намери най-краткия път от даден възел до всички останали в ориентиран теглови граф с алгоритъм на Дийкстра
- *Целева паралелна компютърна платформа*
  - клъстер от работни станции
- *Програмен модел*
  - плосък (flat)
- *Имплементация*
  - MPI

Най-кратък път в граф

- Алгоритъм на Дийкстра
  - намира най-кратък път между два върха в граф или най-кратки пътища от един връх до всички останали
  - прилага се за ориентиран теглови граф с неотрицателни стойности на теглата
  - подобен на алгоритъма на Прим

Най-кратък път в граф

- Алгоритъм на Дийкстра
  - съхранява се множество от възли, за който вече е определен най-кратък път
  - на всяка итерация към това множество се добавя възел, който е най-близък до възел в текущо определения най-кратък път



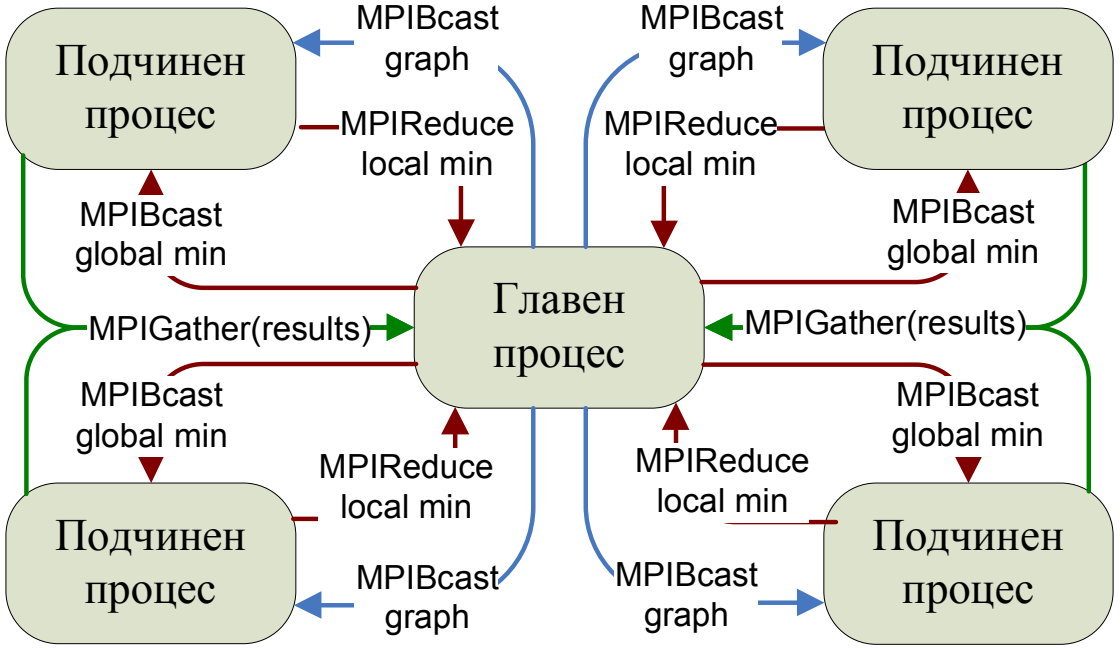
Най-кратък път в граф

- Паралелен алгоритъм на Дийкстра
  - използва се декомпозиция по данни
  - всеки процес изпълнява едни и същи действия като работи над част от върховете в графа
    - при граф с n върха и p процеса всеки процес обработва n/p от върховете в графа
  - всеки процес изчислява минимално разстояние до вече обходените възлите

Най-кратък път в граф

- Модел на комуникация между процесите

- алгоритмични парадигми
  - главен подчинен
  - SPMD



Най-кратък път в граф

- Главният процес изпълнява следните действия
  - разпраща на всички подчинени процеси графа (функция MPI_Bcast())
  - работи като подчинен процес, извършвайки част от изчисленията за определения му блок данни
  - събира резултатите от изчисленията за локално избраните възли (функция MPI_Reduce() с операция MPI_MINLOC)
  - изпраща глобалния минимален възел на всички останали процеси за да го маркират като обходен
  - събира крайните резултати от всички подчинени процеси (функция MPI_Gather())
  - извежда резултата за намерения път след приключване на алгоритъма

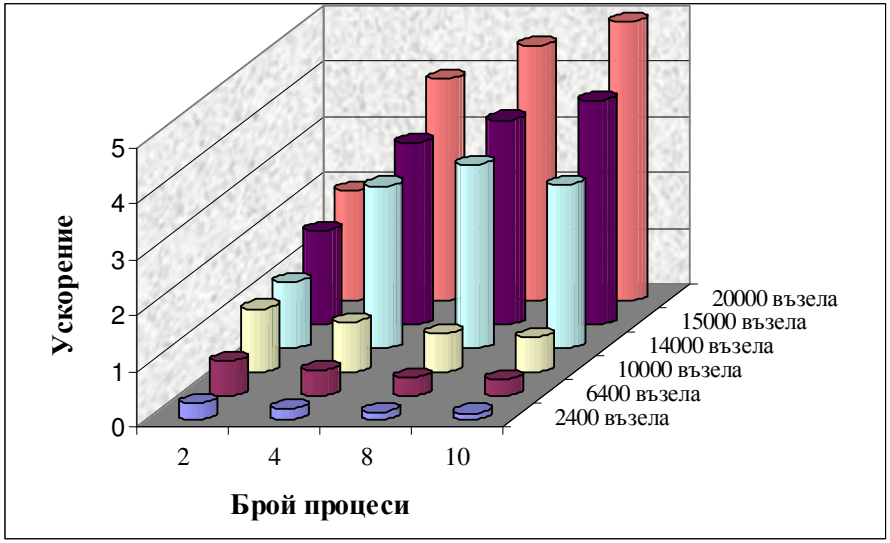
Най-кратък път в граф

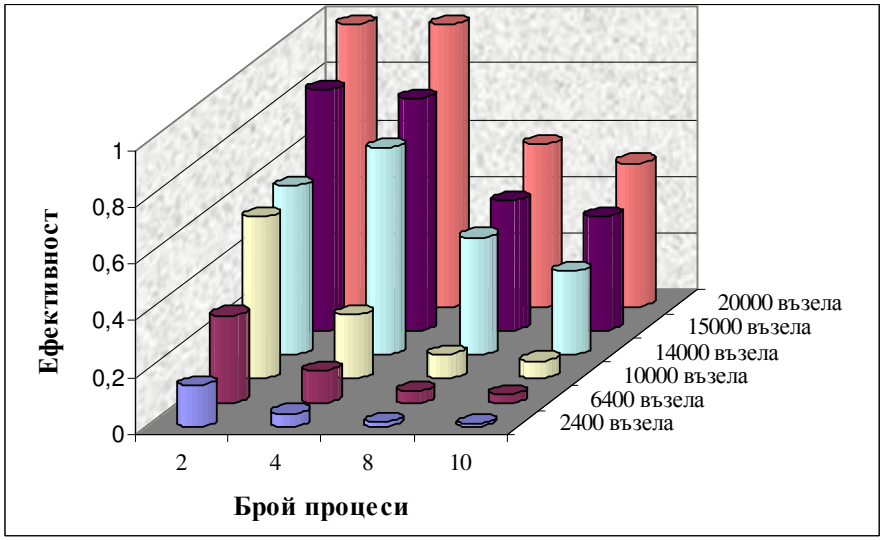
- Всеки подчинен процес извършва следните действия
  - получава от главния процес графа, който ще се обработва
  - извършва изчисления за част от възлите в графа
  - изпраща намерения локален минимум на главния процес
  - получава от главния процес глобалния минимум за текущата итерация и маркират съответния възел като обходен
  - изпраща крайните резултати за намерения път на главния процес

- Вместо двете функции MPI_Reduce() и MPI_Bcast() може да се използва функцията за глобална комуникация MPI_Allreduce()

Най-кратък път в граф

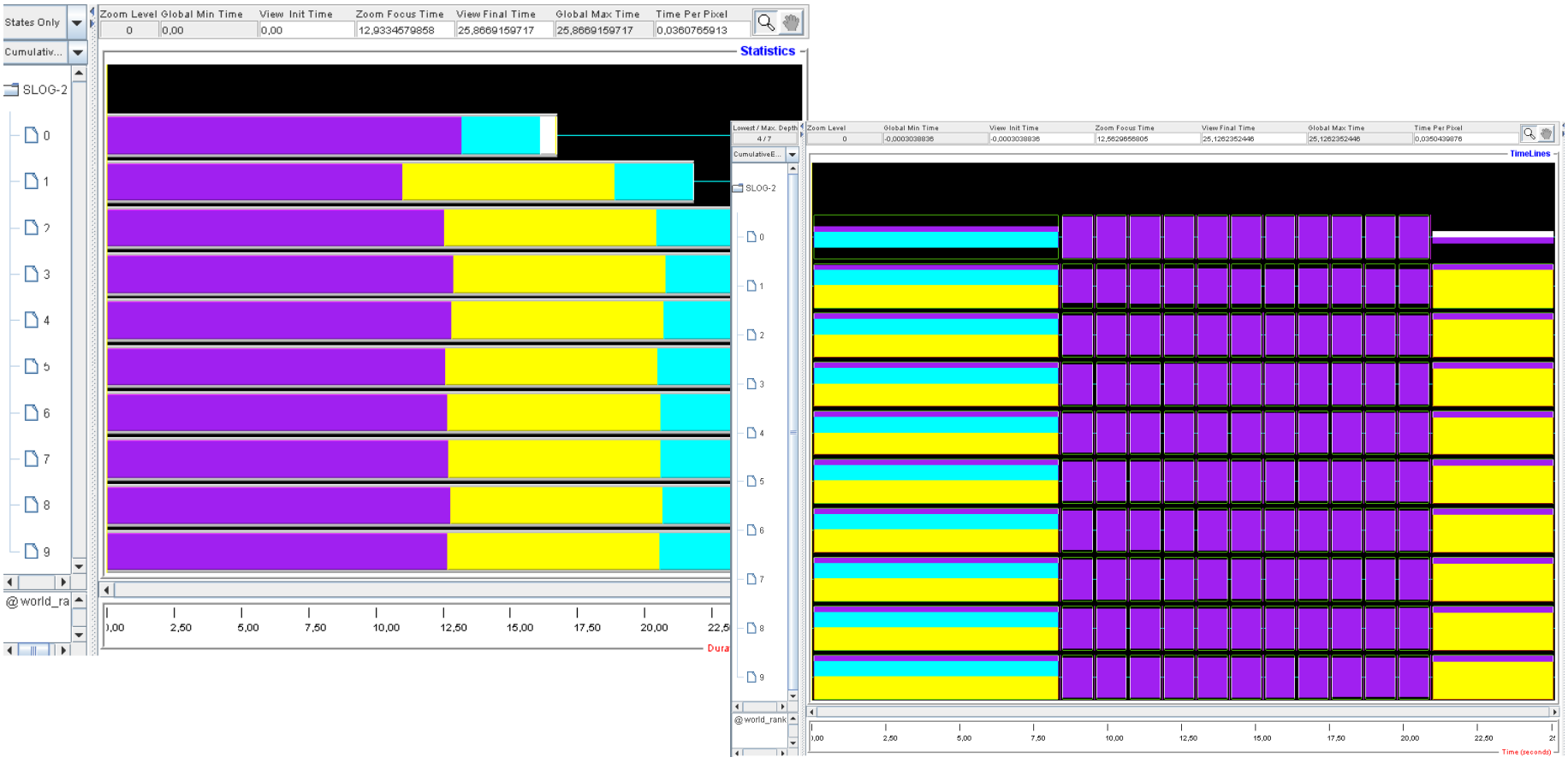
- Производителност
  - почти линейно ускорение
  - висока ефективност на използване на процесорите
  - добро мащабиране





Най-кратък път в граф

- Профил на комуникациите и диаграма на Гант



КРАЙ