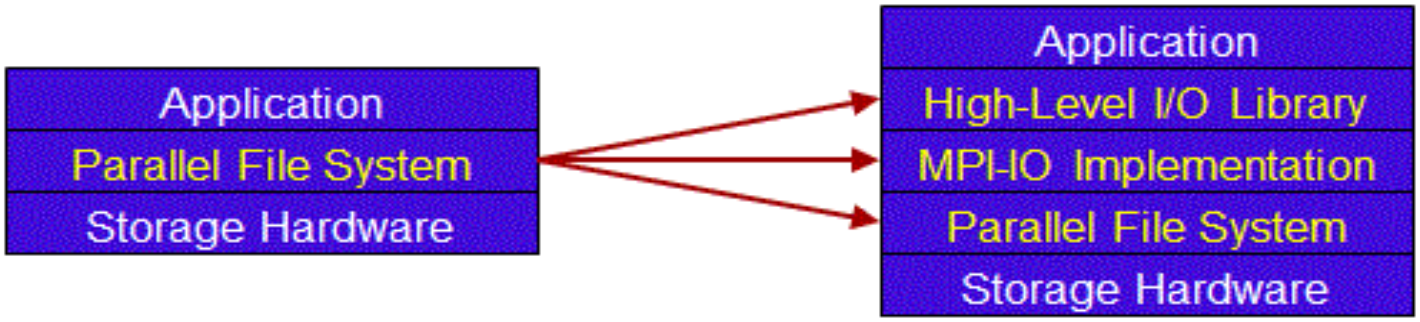
МЕТОДИ И ТЕХНИКИ ЗА ПАРАЛЕЛЕН ВХОД/ИЗХОД ЧАСТ 1

ПРОФ. ПЛАМЕНКА БОРОВСКА

ПАРАЛЕЛЕН ВХОД/ИЗХОД

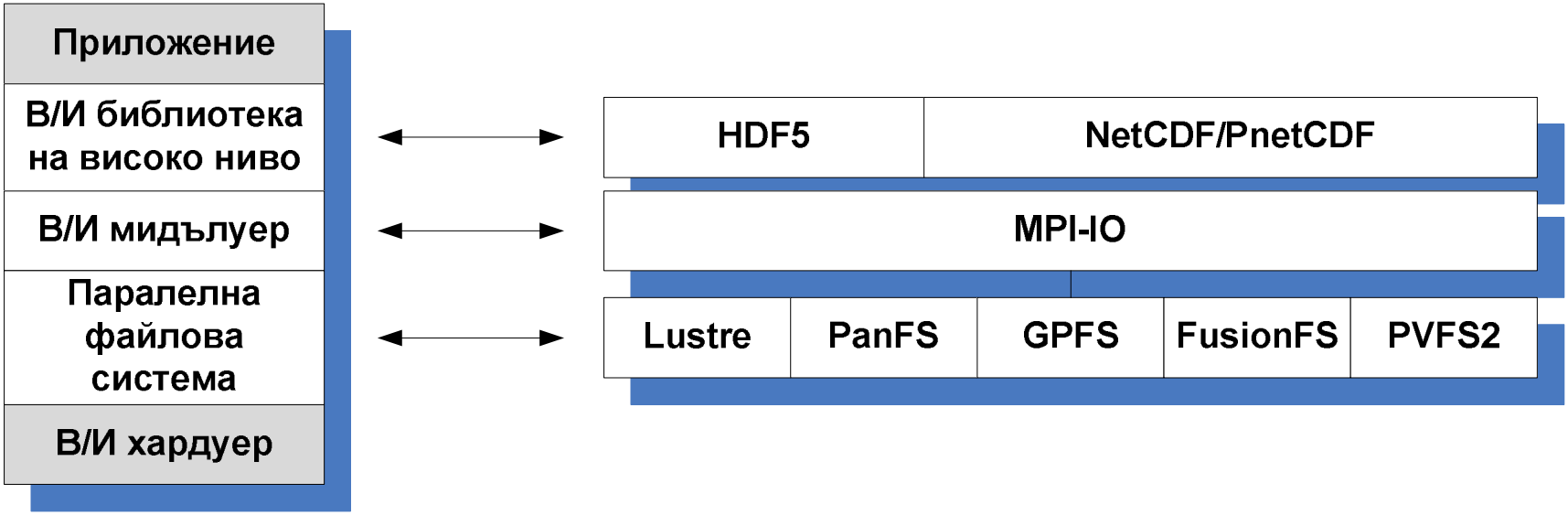
- Паралелният вход-изход означава изпълнението на множество входно-изходни операции едновременно, като например едновременния изход към устройства за съхранение на данни и визуализиращи устройства.
- Конкретен случай е паралелното записване на данни на диск; когато файлът с данните е разпръснат измежду множество дискове, например в RAID масив, в един от дисковете може да се съхранява няколко части от данните по едно и също време, като по този начин се постига по-висока скорост на запис, отколкото с едно устройство.
- Но паралелният вход-изход може да бъде труден за координация и оптимизация. Той е доста тежка задача за създателите на приложенията, тъй като те трябва да ръководят и следят цялото движение на данните.
- С течение на времето се появява друго решение пред специалистите, а именно използването на междинните слоеве.
- По този начин разработчиците на приложения трябва да работят само с интерфейс от високо ниво, който се построява на върха на софтуерния стек и продължава чак до хардуерното ниво

ПАРАЛЕЛЕН ВХОД/ИЗХОД



- Входно-изходната библиотека от високо ниво свързва абстрактно приложенията в структуриран преносим файлов формат (HDF5, NetCDF).
- Средният слой отговаря за достъпа на множество процеси (MPI-IO).
- Паралелната файлова система поддържа логическото файлово пространство и осигурява ефективен достъп до данните (Lustre, PVFS, GPFS).

В/И софтуерен стек



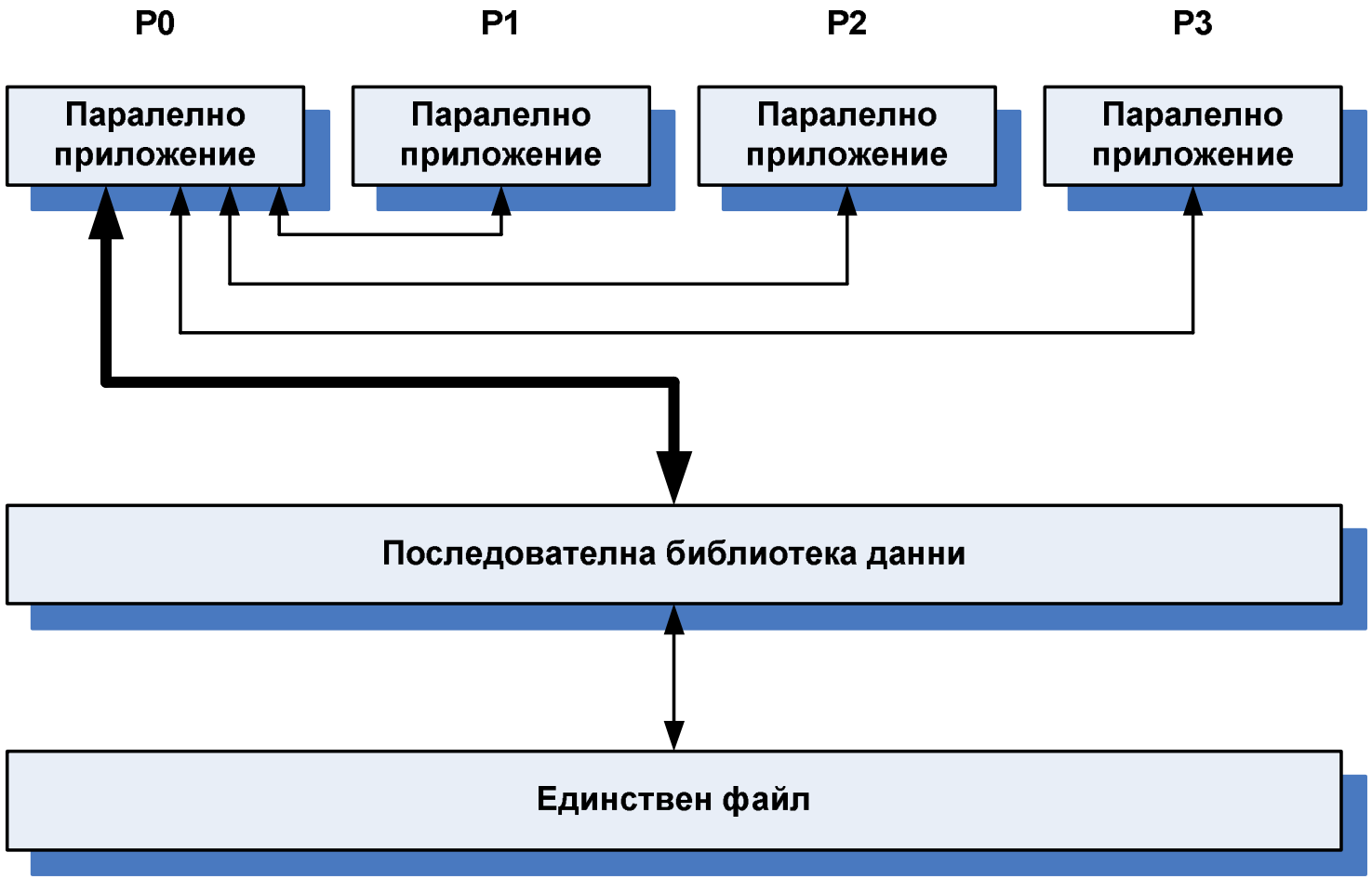
Абстрактен В/И софтуерен стек при научните изчисления

Компоненти на най-популярния В/И софтуерен стек

Техники за високопроизводителен ВХОД/ИЗХОД

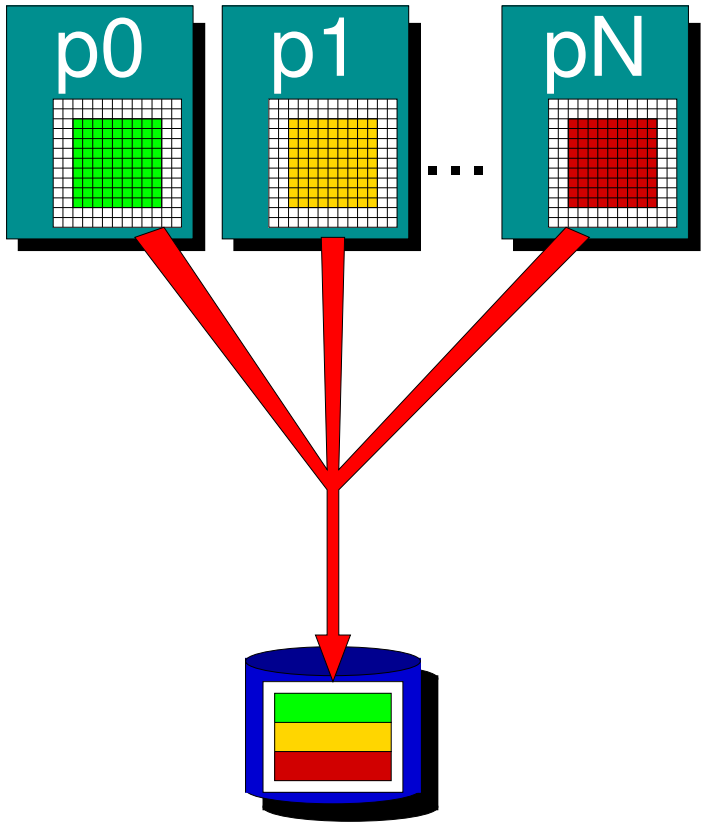
- *В/И интерфейси на ниско ниво, като Unix I/O, третират файловете като последователност от байтове*.
- Научните приложения управляват данните на по-високо ниво на абстракция, при което потребителите могат директно да четат/записват данните като *комплексни структури вместо потоци от байтове*, и всички типове информация, вкл. метаданните, се подават автоматично.
- Приложения, които, в общия случай, се изпълняват на няколко платформи, изискват *преносимост на данните*, така че, данните, генерирани от една платформа, да могат да бъдат използвани на друга платформа без трансформиране.
- Необходимо е да се осигури и *паралелен достъп до данните*, тъй като повечето от научните приложения се изпълняват в паралелни изчислителни среди.

ПОДХОД 1:



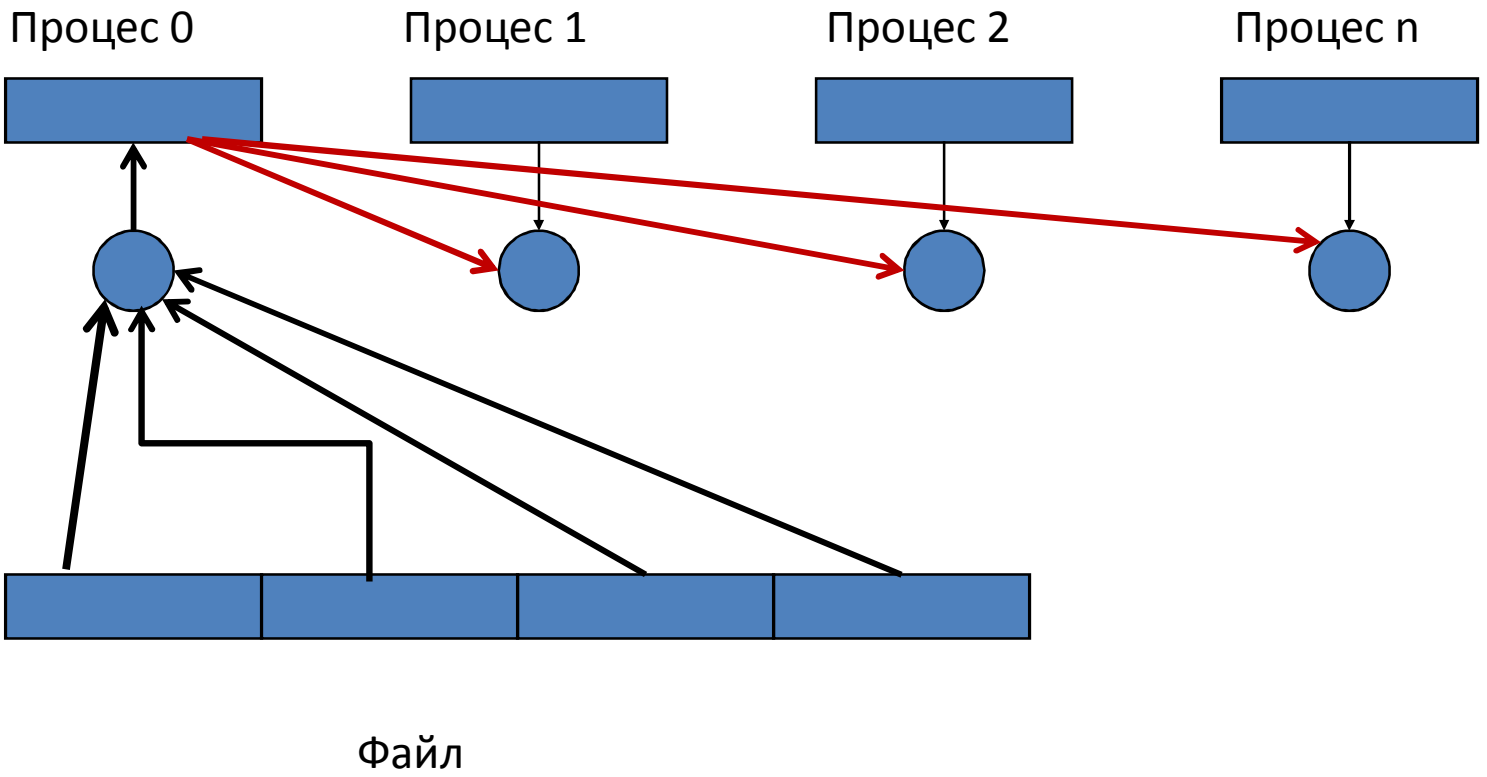
Използване на последователен приложен програмен интерфейс за достъп до единствен файл посредством един процес

ПОСЛЕДОВАТЕЛЕН ДОСТЪП



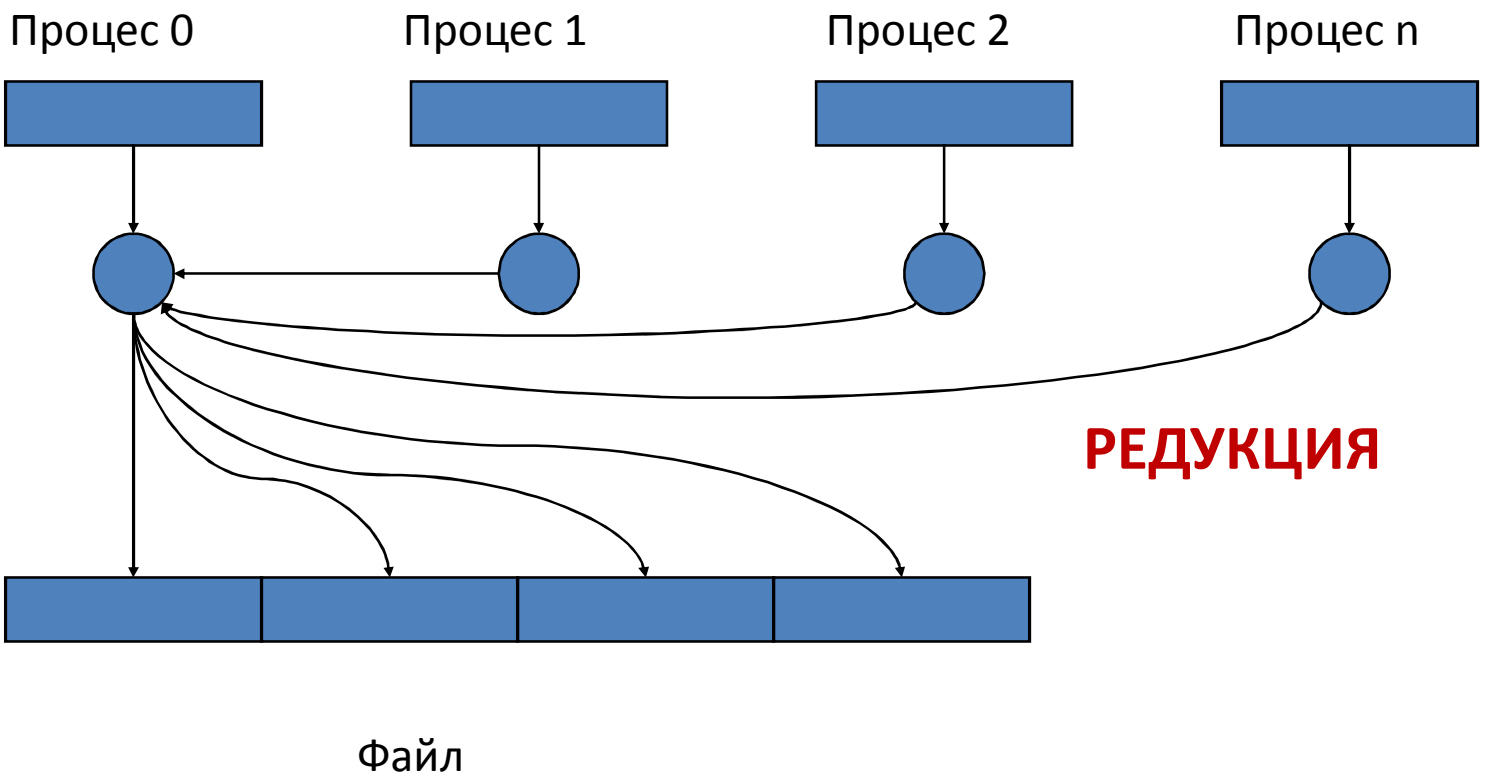
Последователен вход в паралелна програма

- Процесът с ранг 0 чете данните от файл, всички процеси получават данните от процес с ранг 0



Последователен изход в паралелна програма

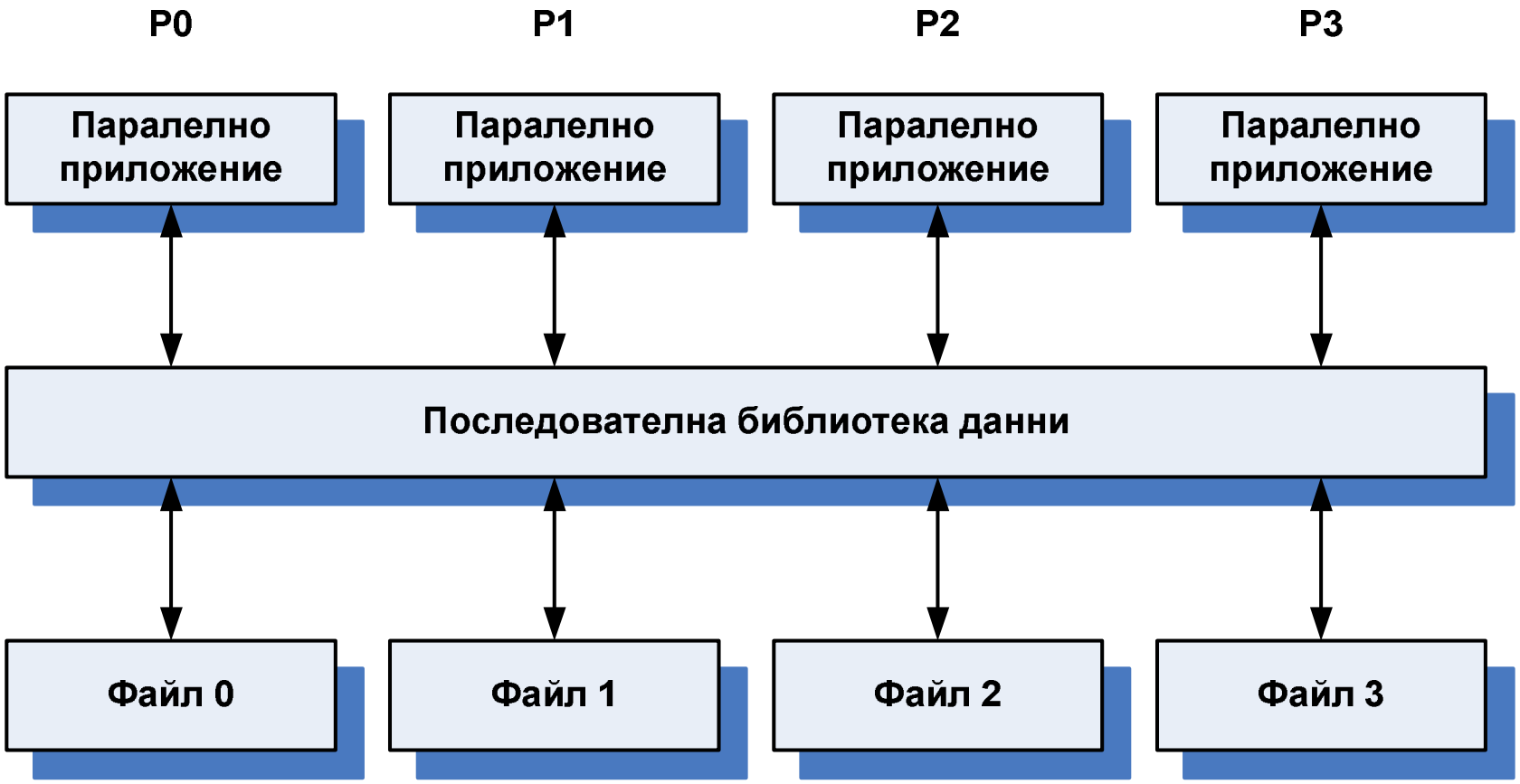
- Всички процеси изпращат данните към процес с ранг 0, и процесът с ранг 0 записва данните във файл



Използване на последователен приложен програмен интерфейс за достъп до единствен файл

- При тази технология за В/И *един процес е отговорен за разпределяне и акумулиране на данните, както и осъществяването на В/И операции с единствен файл като се използва последователен приложен програмен интерфейс (API).*
- Заявките за В/И на всички останали процеси се предават към и осъществяват от този единствен процес.
- Основният недостатък на този подход е, че натоварването на един единствен процес с всички В/И операции лесно може да създаде „тясно място” на В/И производителност, а също така и да ограничи сериозно пропускателната способност на главната памет.

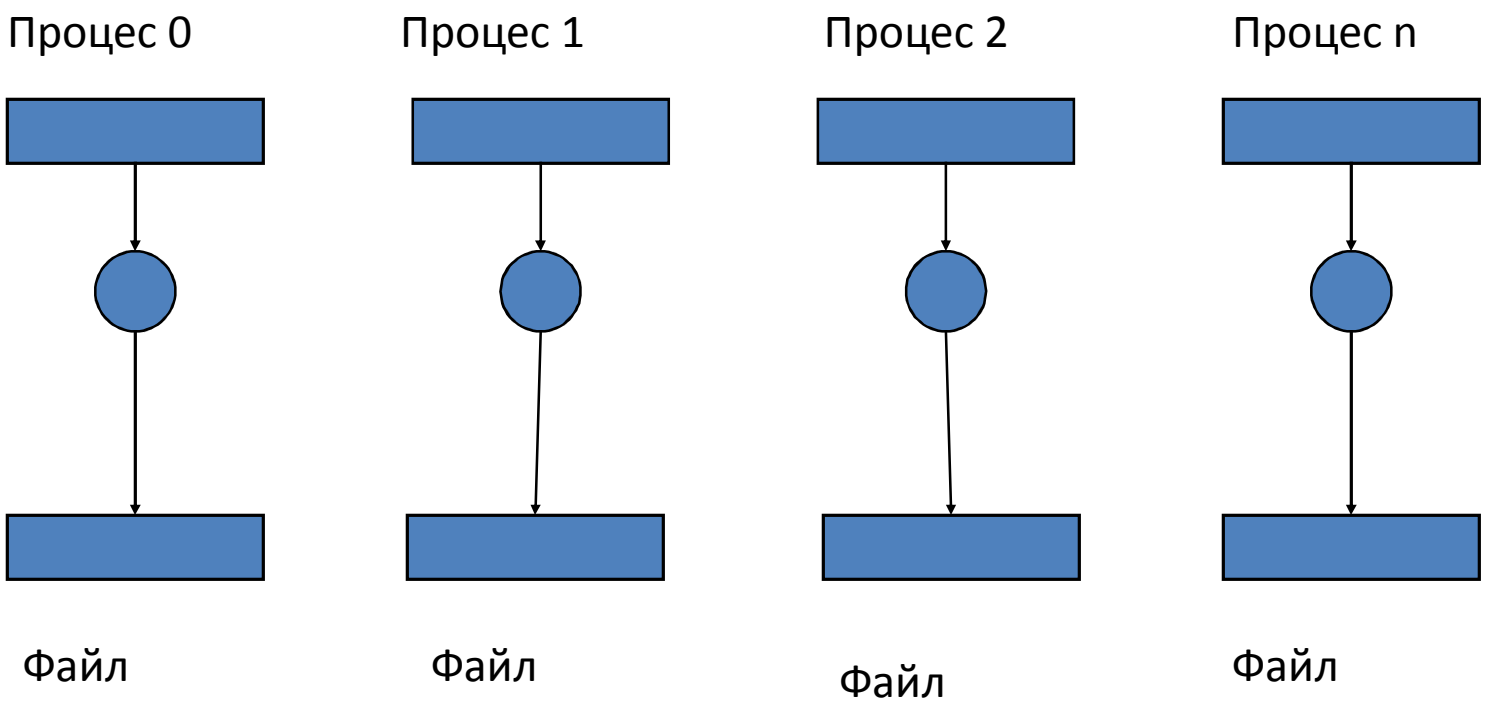
ПОДХОД 2: Използване на последователен приложен програмен интерфейс за едновременен и независим достъп до множество файлове



Използване на последователен приложен програмен интерфейс за едновременен и независим достъп до множество файлове

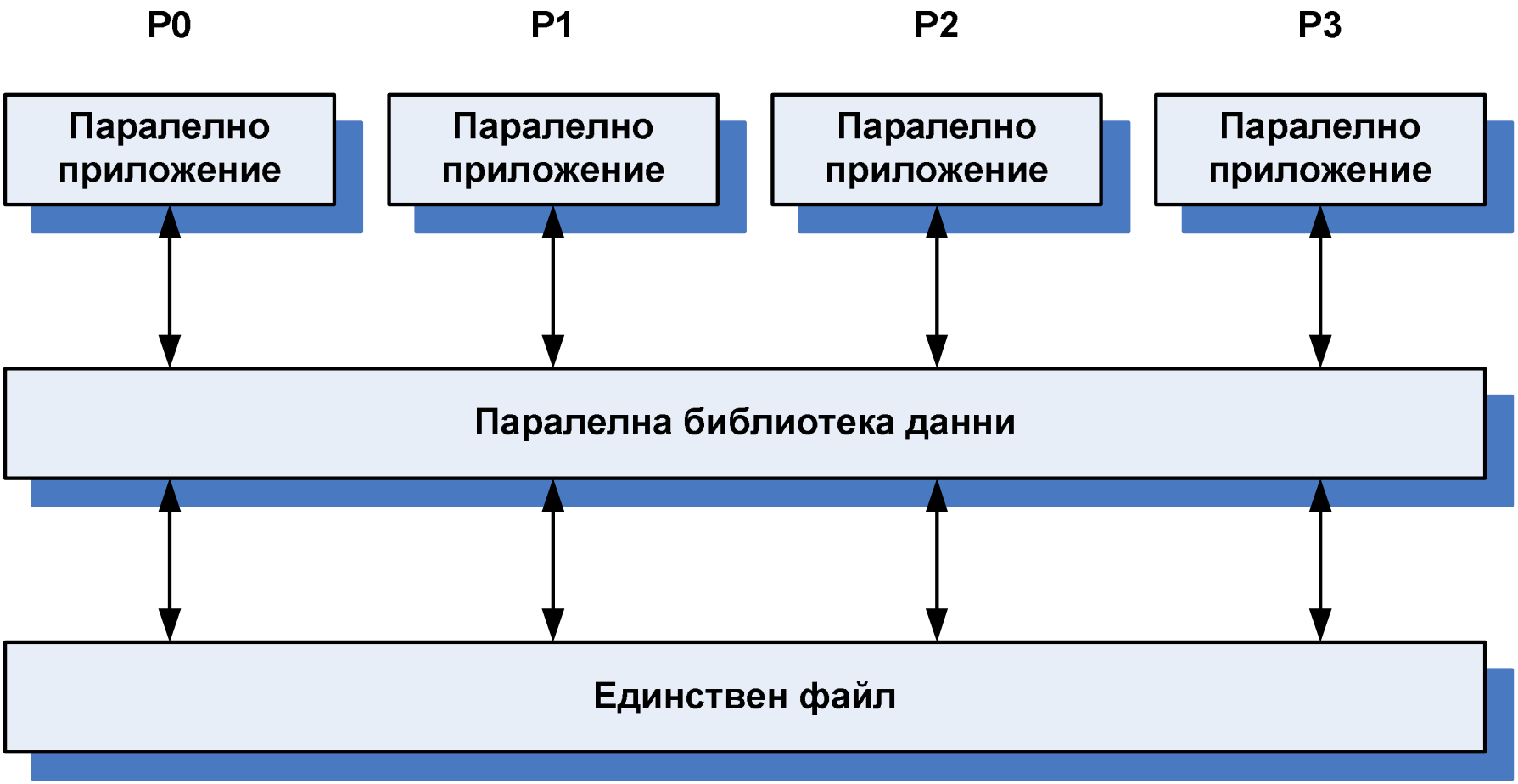
- Всички В/И операции се изпълняват паралелно върху различни файлове (по един файл на процес).
- При този подход, наборът данни е разпределен между множество файлове и управлението на данните е усложнено.
- Този сценарий затруднява да се постигне основната цел на библиотеките, а именно, лесната интеграция и управление на данните.

ПОДХОД 2: Използване на последователен приложен програмен интерфейс за едновременен и независим достъп до множество файлове



- Всеки процес записва данните в отделен файл
- Голям брой малки файлове, бавно при голям брой процеси

ПОДХОД 3: Паралелен приложен програмен интерфейс



Използване на паралелен приложен програмен интерфейс за едновременен и независим достъп до единствен файл колективно или в сътрудничество

- При третия подход се използва паралелен приложен програмен интерфейс със семантика за паралелен достъп и оптимизирана паралелна имплементация на В/И операции.
- При този подход всички процеси осъществяват паралелни В/И операции до единствен файл.
- В резултат, потребителят се освобождава от тънкостите на В/И и осигурява повече възможности за оптимизация на В/И операции.
- Този принцип на дизайн е преобладаващ при съвременните бибиотеки за научни данни.

Преносими файлови формати NetCDF и паралелен NetCDF

- *NetCDF (Net Common Data Format - Мрежов общ формат данни)* е разработен от Unidada Program Center.
- Осигурява общ метод за достъп до структурираните набори данни в дисковете на приложенията.
- Приложенията на науката за атмосферата, например, използва *NetCDF* за съхраняване на различни типове данни, които обхващат единични точки на наблюдение, времеви серии, регулярни мрежи, сателитни или радарни образи.
- Много организации, като например, общността за моделиране на климата, разчитат на стандарта на *NetCDF* за достъп до данни от дисковите масиви.

NetCDF

- *NetCDF* съхранява наборите данни под формата на масиви, съдържащи измерения, променливи, и атрибути.
- *Физически, файлът с набора данни, е разделен на две части: заглавна част и масиви данни*.
- *Заглавната част съдържа мета данни* – цялата информация за измеренията, атрибутите и променливите (но не и стойностите на променливите), докато *секцията данни съдържа масиви от стойностите на променливите („сурови данни”).*
- *Масиви с фиксиран размер се съхраняват в непрекъснати област от специфицирано отместване във файла, докато масиви с променлив размер се съхраняват в края на файла под формата на редуващи се записи*, които нарастват заедно по протежението на общо неограничено измерение.

Операциите при *NetCDF* се разделят на 5 категории:

Тип на функцията	Описание
Функции за набора данни	Отваряне/затваряне/създаване на набор данни Синхронизиране на набор данни
Функции за определяне на режима	Дефиниране на измеренията на наборите данни и променливите
Функции за атрибутите	Управление на прибавянето, промяната и четенето на атрибутите на наборите данни
Функции за справки	Връща мета данните на набора данни: dim (id,name,len); var(name, ndims, shape, id)
Функции за достъп до данни	Осигурява възможност за четене/запис на данни по един от 5 метода за достъп до файла: *(1) единична стойност (single value), (2) цял масив (whole array), (3) подмасив (subarray), (4) стъпкови извадки от масив (strided subarray), (5) картирани стъпкови извадки от масив (mapped strided subarray)*

Типичната последователност от операции при записа на нов набор данни при NetCDF е:

1. Създаване на набора данни;
2. Дефиниране на измеренията, променливите и атрибутите;
3. Запис на стойностите на променливите;
4. Затваряне на набора данни.

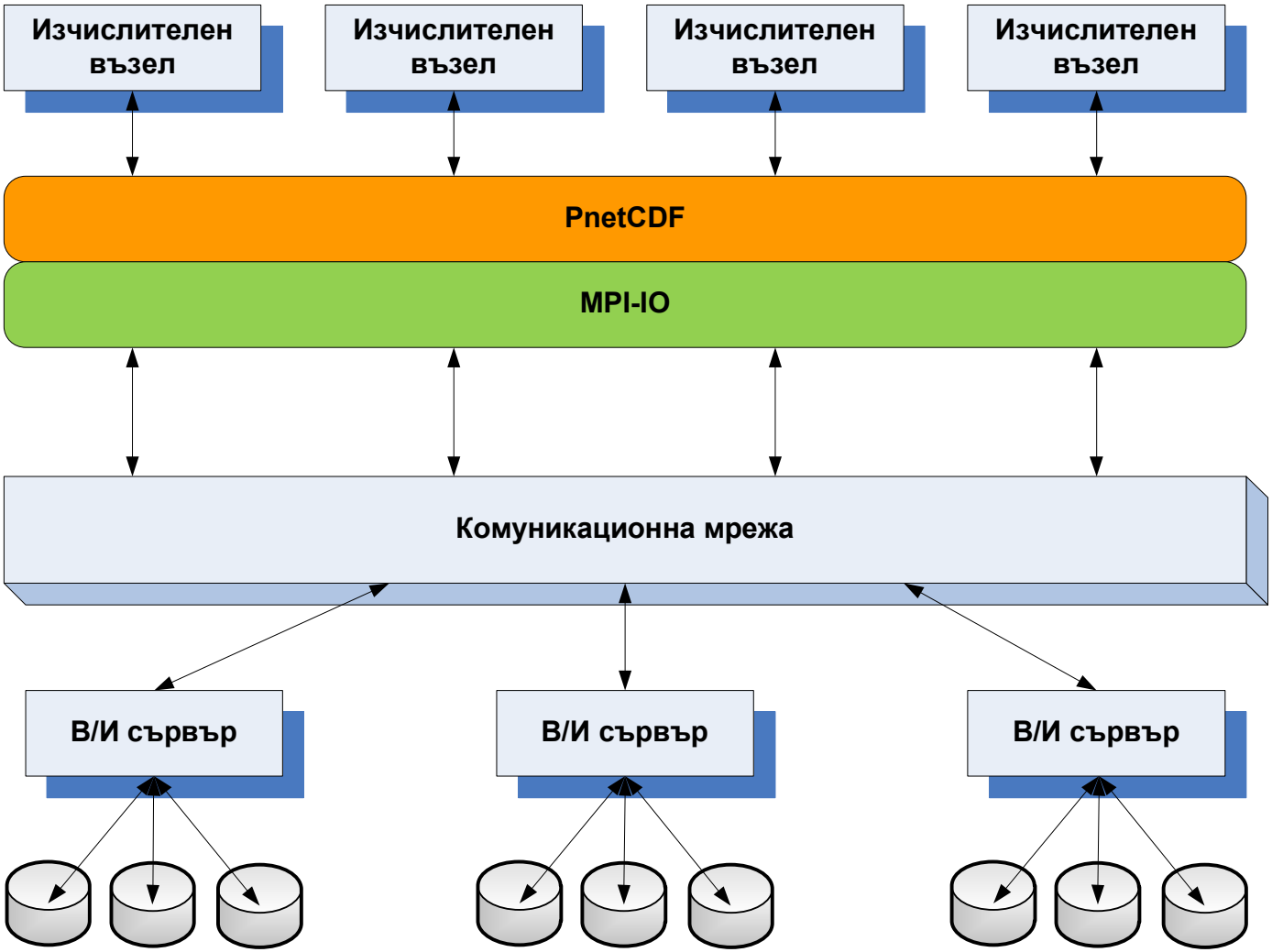
Типичната последователност от операции при четенето на съществуващ набор данни NetCDF е:

1. Отваряне на набора данни;
2. Справка за измеренията, променливите и атрибутите;
3. Четене на стойностите на променливите;
4. Затваряне на набора данни.

Паралелната версия PNetCDF

- Първоначално, приложният програмен интерфейс на NetCDF е проектиран за последователен достъп до данните, без паралелна семантика.
- *Паралелната версия PNetCDF е разработена съвместно от Northwestern University и Argonne National Laboratory и осигурява паралелен приложен програмен интерфейс за достъп до файловете NetCDF със значително по-висока производителност.*
- Библиотеката NetCDF е изградена върху MPI-IO, като по този начин осигурява възможност потребителите да се възползват от няколко популярни оптимизации в MPI-IO-базираните имплементации, а именно „пресяването" на данните и стратегиите за двуфазен В/И при ROMIO.

Архитектура на PNetCDF



PnetCDF

- *PnetCDF се изпълнява като библиотека между потребителското приложение и файловата система.*
- Паралелните заявки netCDF се обработват на потребителските изчислителни възли и след оптимиране, паралелните заявки за В/И се предават на библиотеката MPI-IO.
- В/И сървъри получават заявките MPI-IO и изпълняват В/И операции с крайните дискове за потребителя.

PnetCDF

- *При PnetCDF файловете се отварят, обработват и затварят от всички процеси- участници в MPI комуникатора*.
- Заглавната част на файла се чете/записва от единствен процес, въпреки че, се кешира по едно копие в локалната памет на всеки процес.
- При отварянето на файл, процесът-корен чете заглавната част на файла и в случай, че тя се модифицира, я записва.
- Функциите за определяне на режима, функциите за атрибутите и функциите за заявките се изпълняват над локалното копие на заглавната част на файла.

PnetCDF

- Функциите за определяне на режима и функциите за атрибутите се изпълняват колективно като се осигуряват еднакви аргументи при прибавянето, отстраняването или промяната на дефиниции, така че, се гарантира локалните копия на хедъра на файла да бъдат еднакви при всички процеси от момента на колективното отваряне на файла до затварянето му.

PnetCDF

- *Паралелизацията на функциите за достъп до данните се постига чрез 2 подмножества приложни програмни интерфейси: high-level API и flexible API*.
- Приложният програмен интерфейс на високо ниво използва функциите за достъп до данни на оригиналния netCDF и улеснява потребителите да преминат към използването на паралелния интерфейс.
- Гъвкавият *API* следва стила на достъп на MPI като осигурява на потребителя възможност да описва прекъснати и разпръснати области в паметта.
- Тези области в паметта се описват като MPI типове.

HDF (Hierarchical Data Format)

- Йерархичният формат на данните HDF представлява *преносим файлов формат и софтуер*, разработен от National Center for Supercomputing Applications (NCSA).
- Предназначен е за съхраняване, извличане, анализ, визуализация и конвертиране на научни данни.
- *Настоящата и най-популярна версия е HDF5*, при която многомерните масиви се записват с допълнителни данни в преносим самоописващ се файлов формат.
- Използва се йерархична структура, която осигурява на приложните програмисти множество опции за организиране на данните в HDF5 файлове.

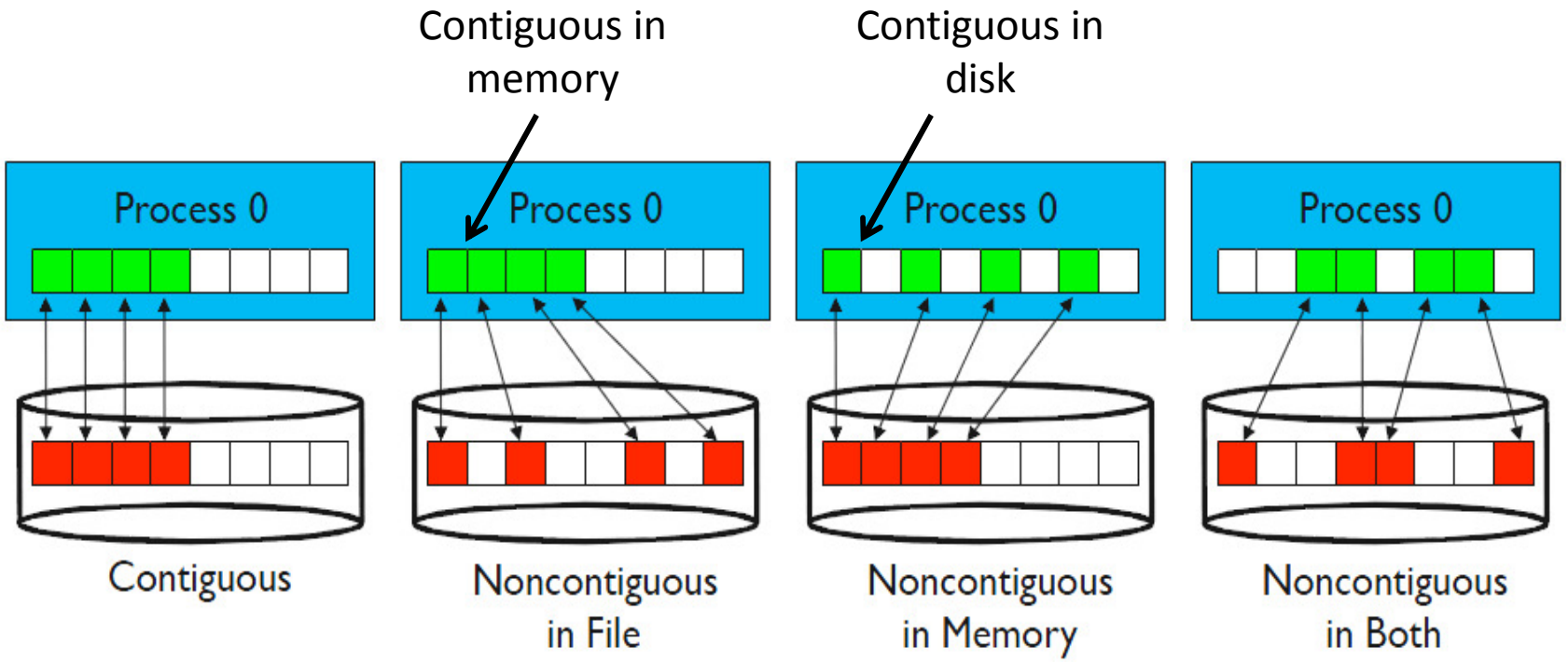
HDF (Hierarchical Data Format)

- *Поддържат паралелен В/И*.
- HDF5 файлове са организирани в йерархична структура, подобна на файловата система на UNIX.
- *В структурата се записват 2 типа първични обекти – групи и набори данни*.
- Групата съдържа инстанции на групи или набори данни, докато наборите данни съдържат многомерни масиви от данни.
- *Физически, наборът данни се записва в 2 части: хедър и масив данни*.
- Размерността на набора данни може да бъде фиксирана или неограничена (разширяема).
- Противно на netCDF, HDF5 поддържа повече от една неограничена димензия в пространството данни.

HDF (Hierarchical Data Format)

- *HDF5 поддържа достъп до избрани селекции в наборите данни като използва т. нар. „хиперплочки”,* техни обединения или списъци от независими плочки.
- *Хиперплочката представлява подмасив или извадки от елементите на масива при фиксирана стъпка на многомерни набори данни*.

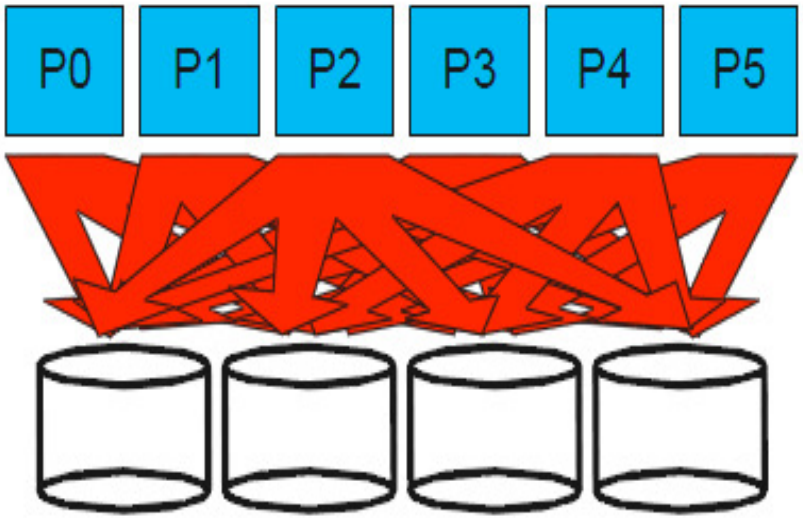
Разпръснатият В/И (noncontiguous I/O) има 3 форми: разпръснатост в паметта, разпръснатост във файла, или разпръснатост и в двете.



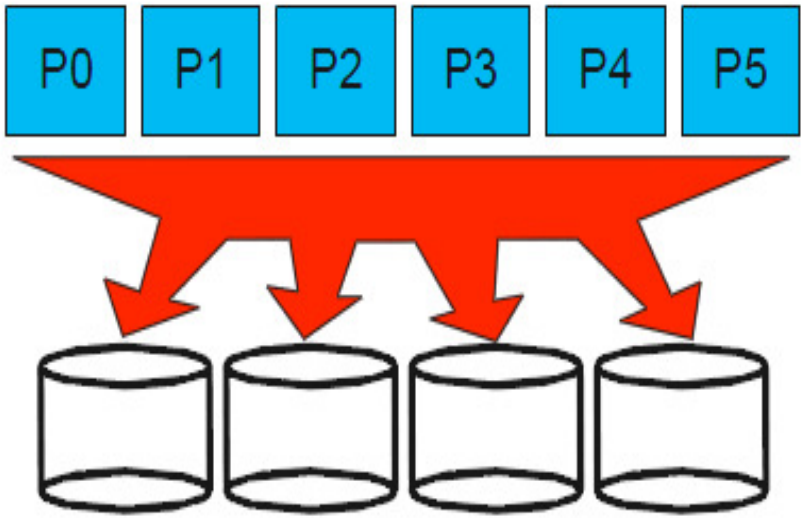
Колективен вход/изход

- Много популярен подход за В/И, базиран на библиотеката MPI-IO, е колективният В/И, който дефинира набор от подпрограми, които осъществяват трансфер на данни към/от външната памет.
- Производните типове данни за паметта и за файловете на колективния В/И интерфейс позволяват съвместни В/И оптимизации при процесите като моделите за достъп на високо ниво се запазват непроменени.
- MPI-IO осигурява възможост за четене и запис в нормален режим, т.е. режим с блокиране, и впоследствие – в асинхронен режим без блокиране, така че, паралелно с обработката на процесите, файловете могат да бъдат четени/записвани във фонов режим.

ФОРМИ НА В/И

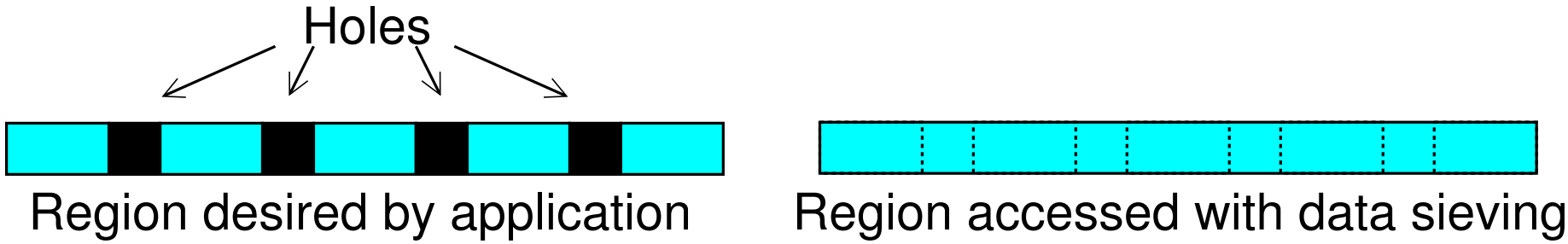


Independent I/O



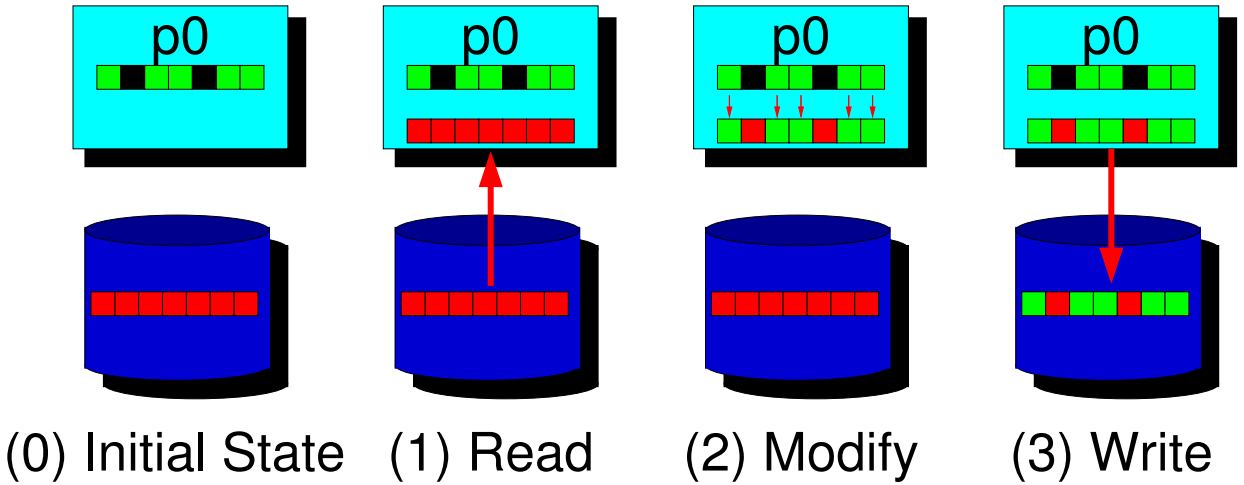
Collective I/O

Пресяване на данни (Data Sieving)



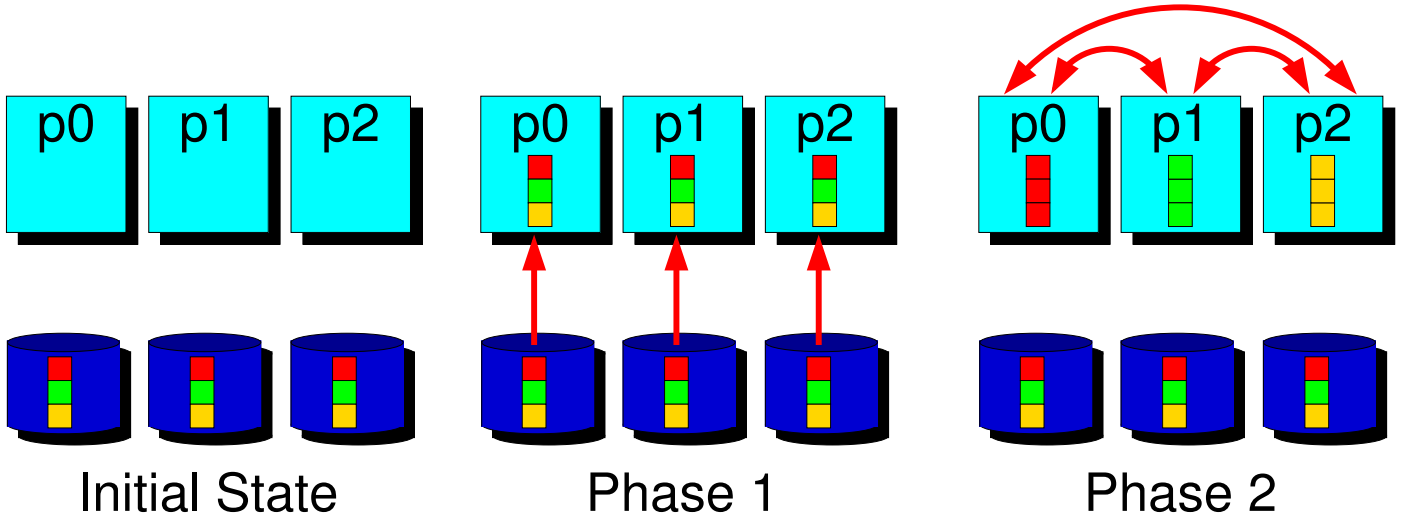
- Data sieving се използва за комбиниране на множество кратки достъпи в един интегриран достъп
- Мотиви:
  1. Отдалечените файлови системи (паралелни или не) имат голяма латентност
  2. Редуцирането на броя операции е от изключително значение
- По приницп е високо ефективна техника, но по-ефективно решение е паралелната файлова система, която поддържа noncontiguous access

Data Sieving Writes



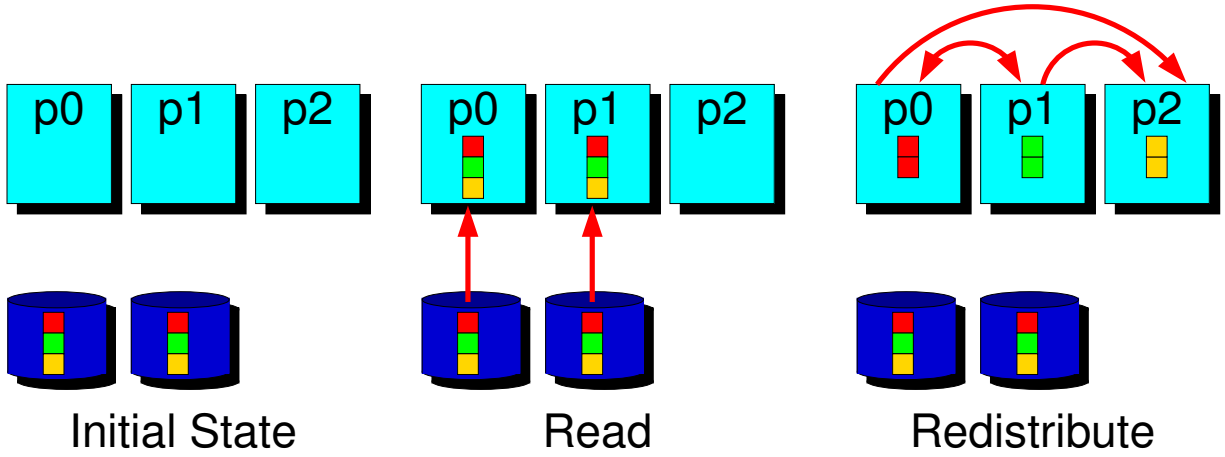
- Използването на техниката data sieving при записи е по-сложна
  - Първо трябва да се прочете целия блок
  - После се правят промените - актуализация
  - Запис на блока на диска
- Изисква “заключване” (locking) във файловата система
  - В резултат може да се получи ефекта “false sharing” припокрит достъп - interleaved access)
  - За предпочитане е да се използват паралелни файлови системи с noncontiguous writes

Двуфазен колективен В/И



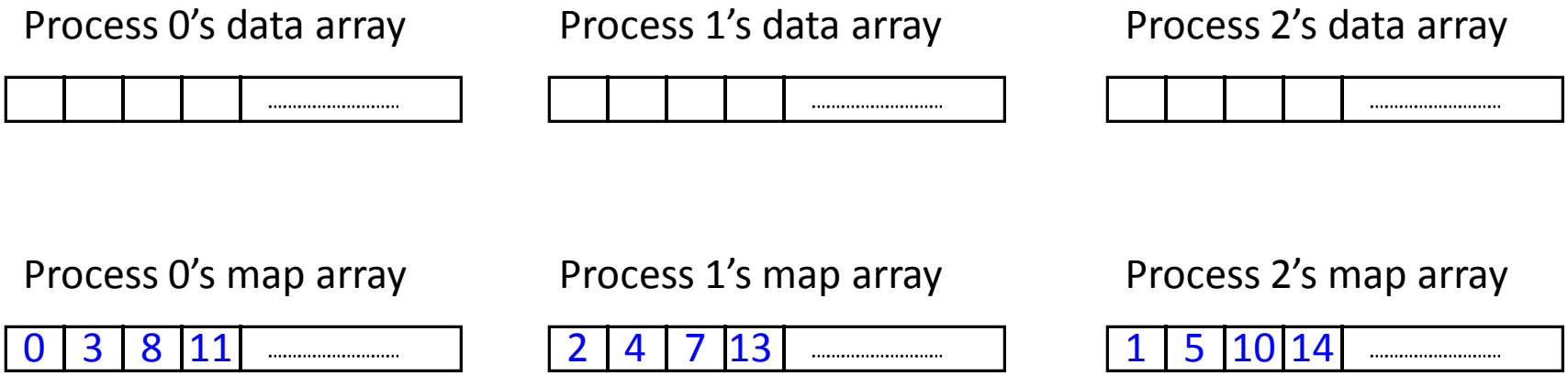
- Проблеми при независими разпръснати (noncontiguous) достъпи
  - Голям брой достъпи до малки данни
  - Независимите четения с пресяване на данните изисква четене на много излишни данни
- Идея: да се реорганизират достъпите с цел съответствие на разположението на данните върху диска
  - Отделните процеси пресяват данните за всички процеси
  - Често се редуцира времето за В/И посредством споделяне на общи блокове
- При втората фаза данните се предават към дестинациите

Агрегация (Aggregation)



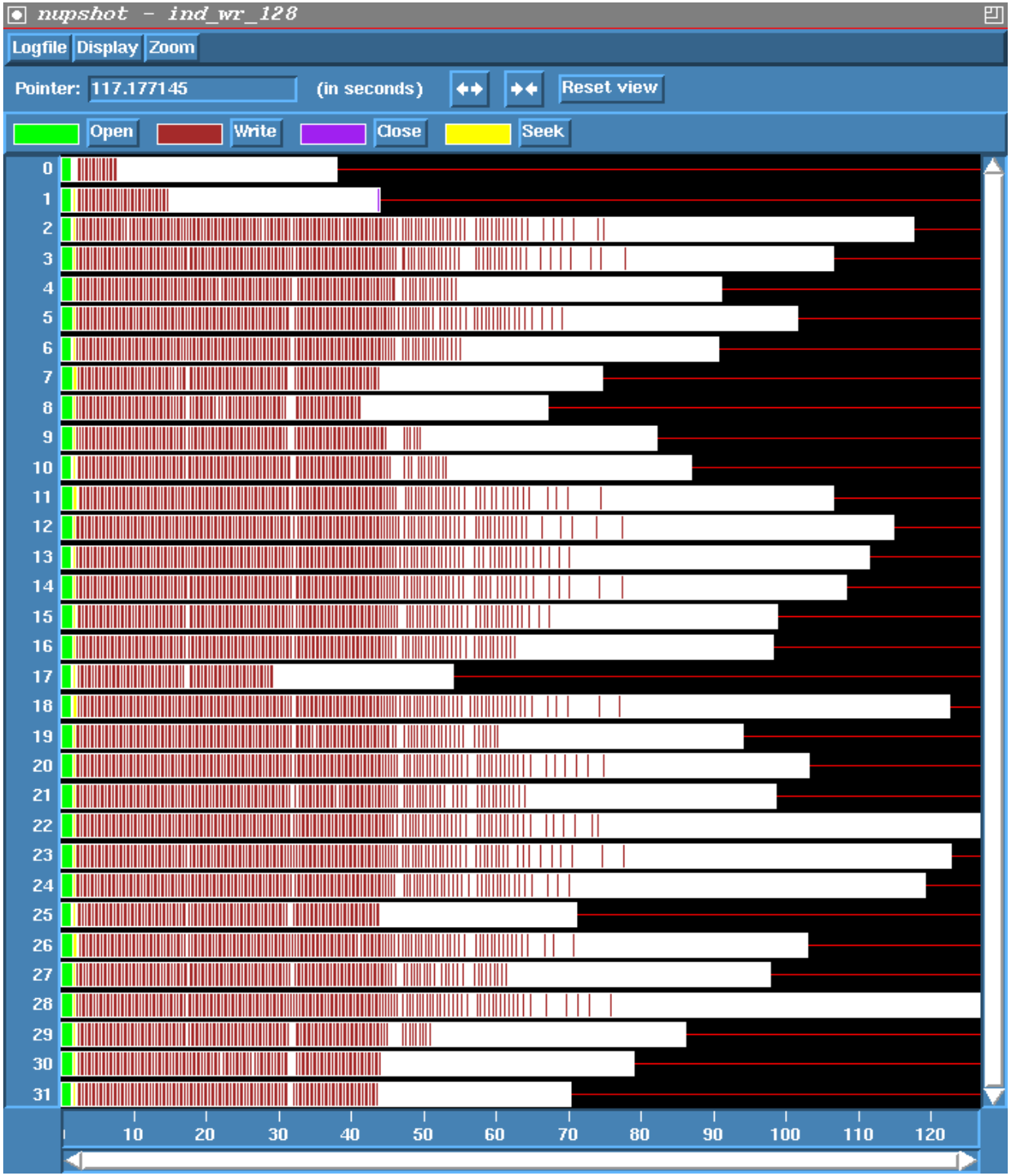
- По-общо приложение на концепцията за трансфер на данни през междинни изчислителни възли
  - Различен брой изчислителни възли могат да осъществяват В/И
  - Може да се приложи при независим В/И
- Може да се прилага при отдалечен В/И, където процесите – агрегатори са на различни системи

Достъп до нерегулярно разпределени масиви

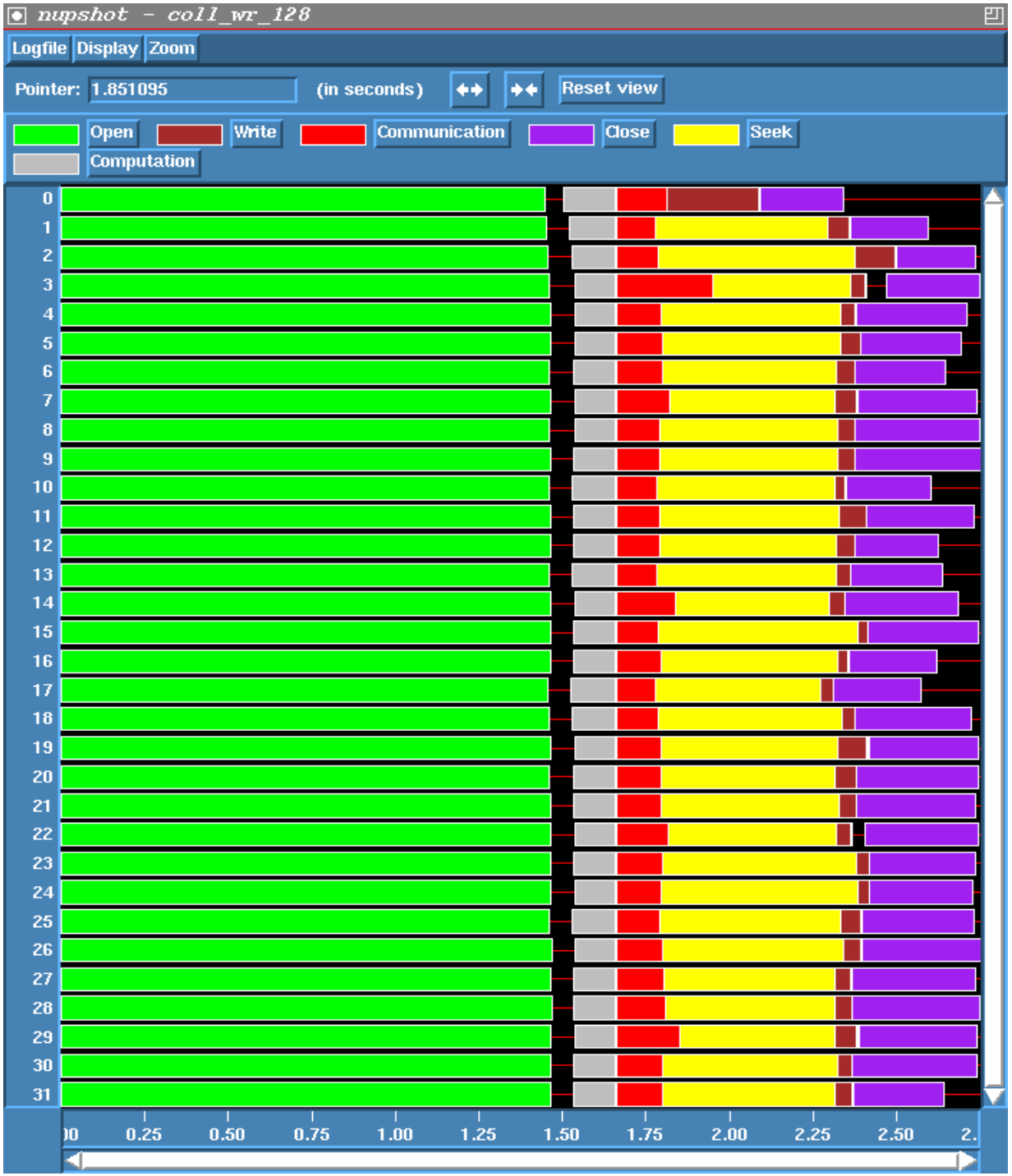


Картата на масива (*map array*) описва разположението на всеки елемент от масива в общия файл

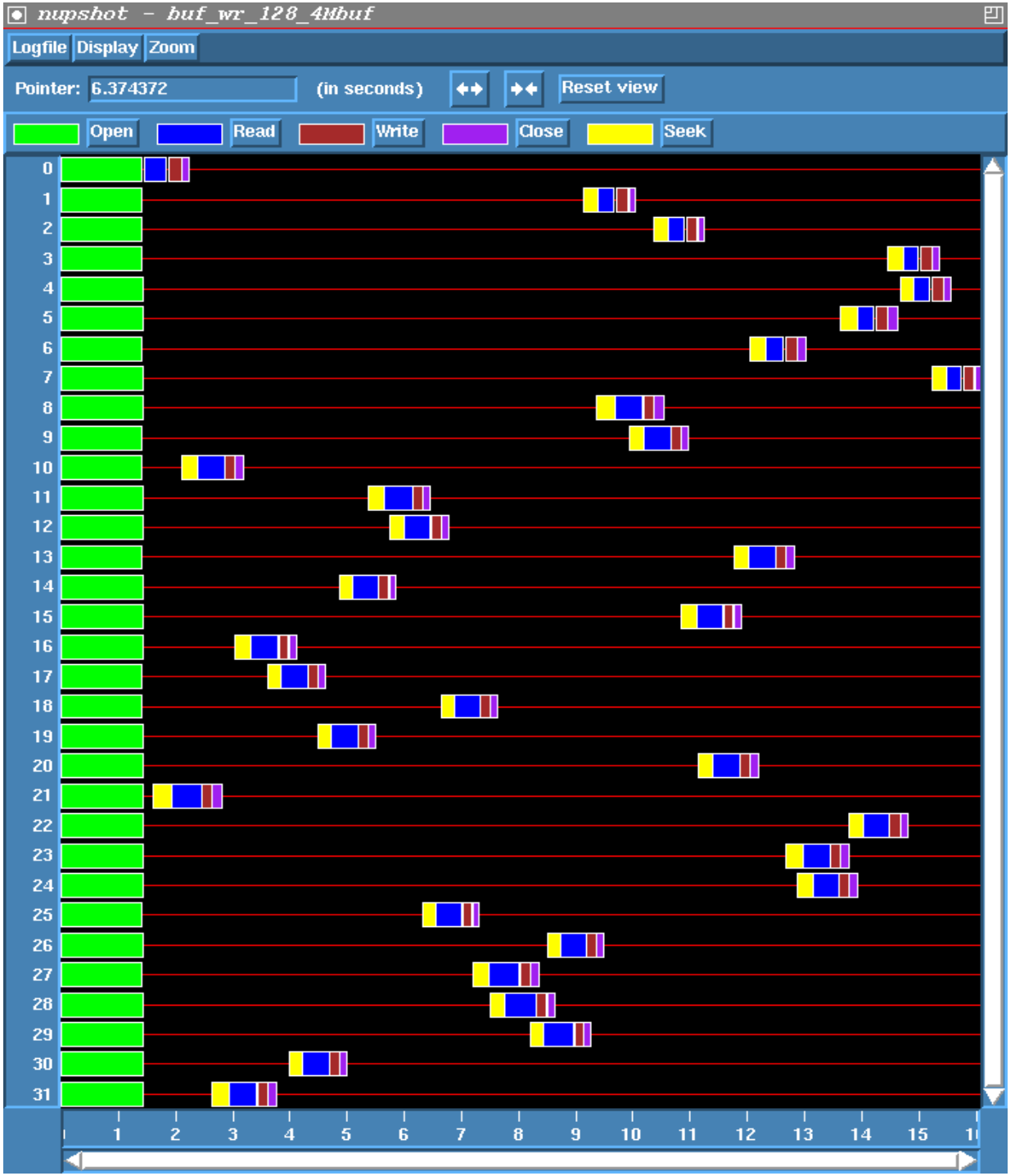
Профил на независими записи на Paragon за 130 s



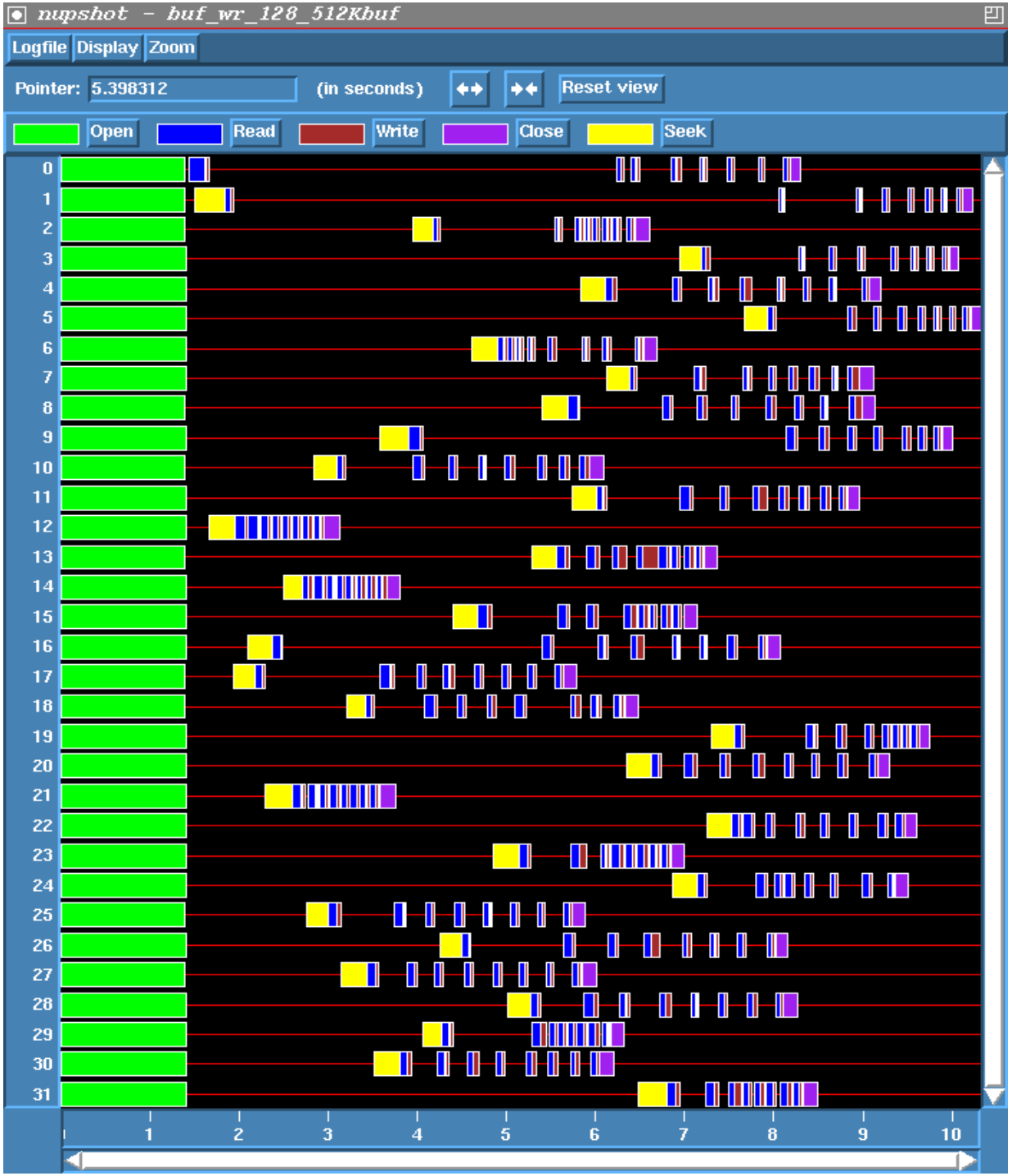
Колективни записи на Paragon



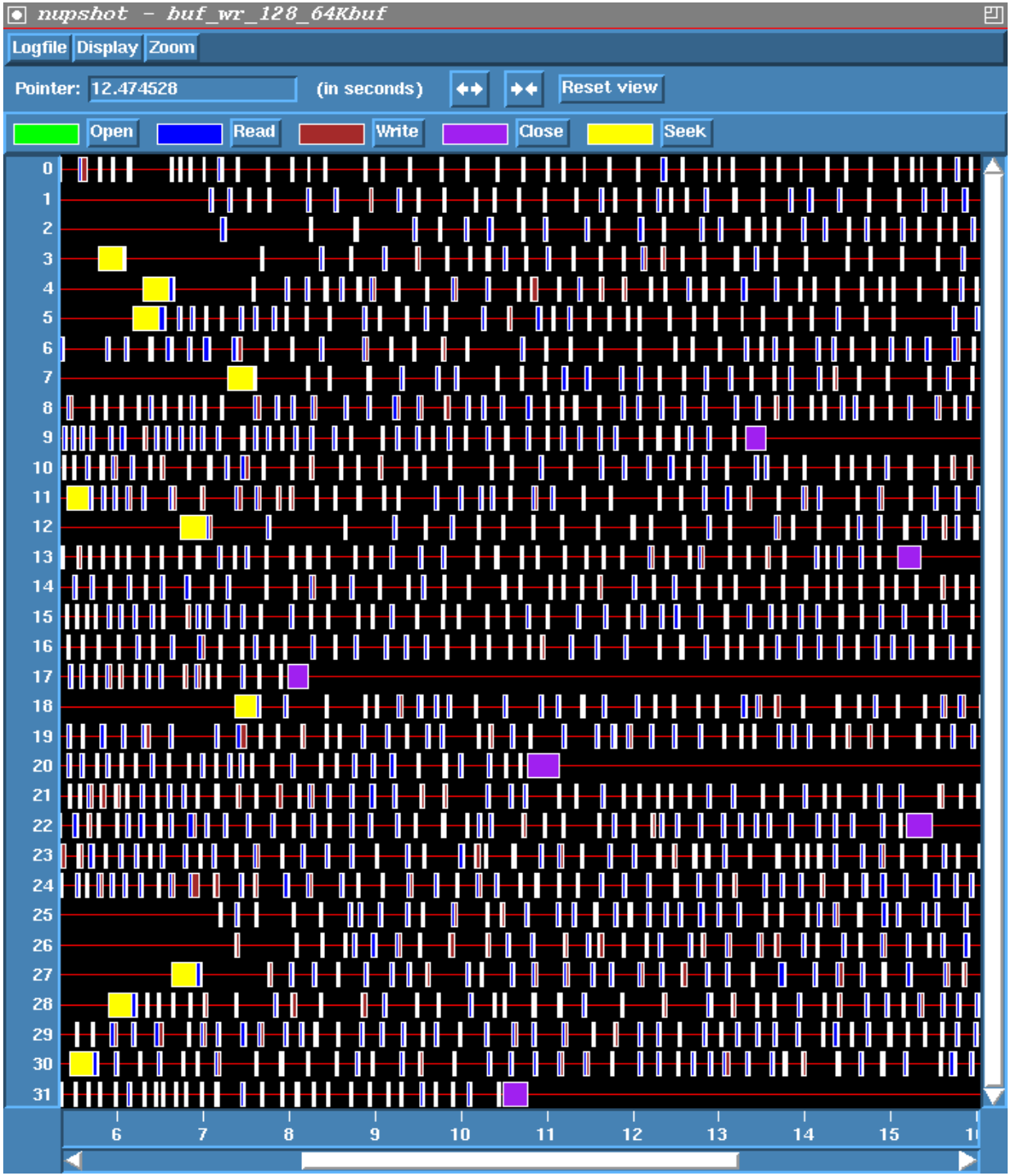
Независими записи с Data Sieving на Paragon – достъп до блокове данни 4 MB и извличане на необходимите данни - изисква lock, read, modify, write, unlock for writes blocks време 16 sec.



Независими записи с Data Sieving на Paragon – достъп до блокове данни 512 KB и извличане на необходимите данни - изисква lock, read, modify, write, unlock for writes blocks време 10.2 sec.



Независими записи с Data Sieving на Paragon – достъп до блокове данни 64 KB и извличане на необходимите данни - изисква lock, read, modify, write, unlock for writes blocks време 21.5 sec.



КРАЙ
ЧАСТ 1