Управление памятью

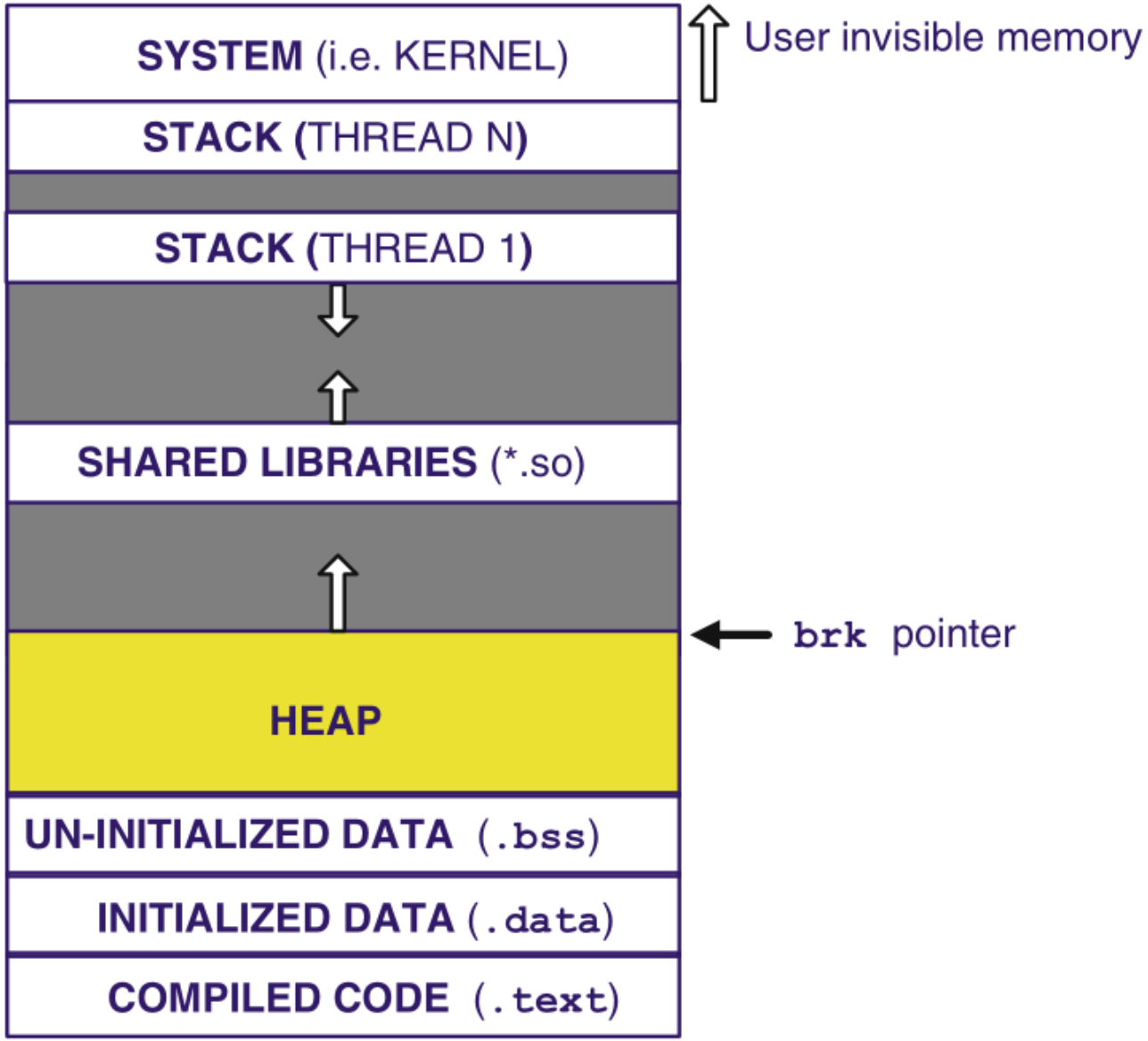
Евгений Иванович Клименков

osisp2019@gmail.com

Белорусский Государственный Университет
Информатики и Радиоэлектроники

2019

Heap


SYSTEM (i.e. KERNEL)
User invisible memory
STACK (THREAD N)
STACK (THREAD 1)
SHARED LIBRARIES (*.so)
brk pointer
HEAP
UN-INITIALIZED DATA (.bss)
INITIALIZED DATA (.data)
COMPILED CODE (.text)

Heap Interface

C style

- void* malloc (size_t size);
- void* calloc (size_t num, size_t size);
- void* realloc (void* ptr, size_t size);
- void free (void* ptr);

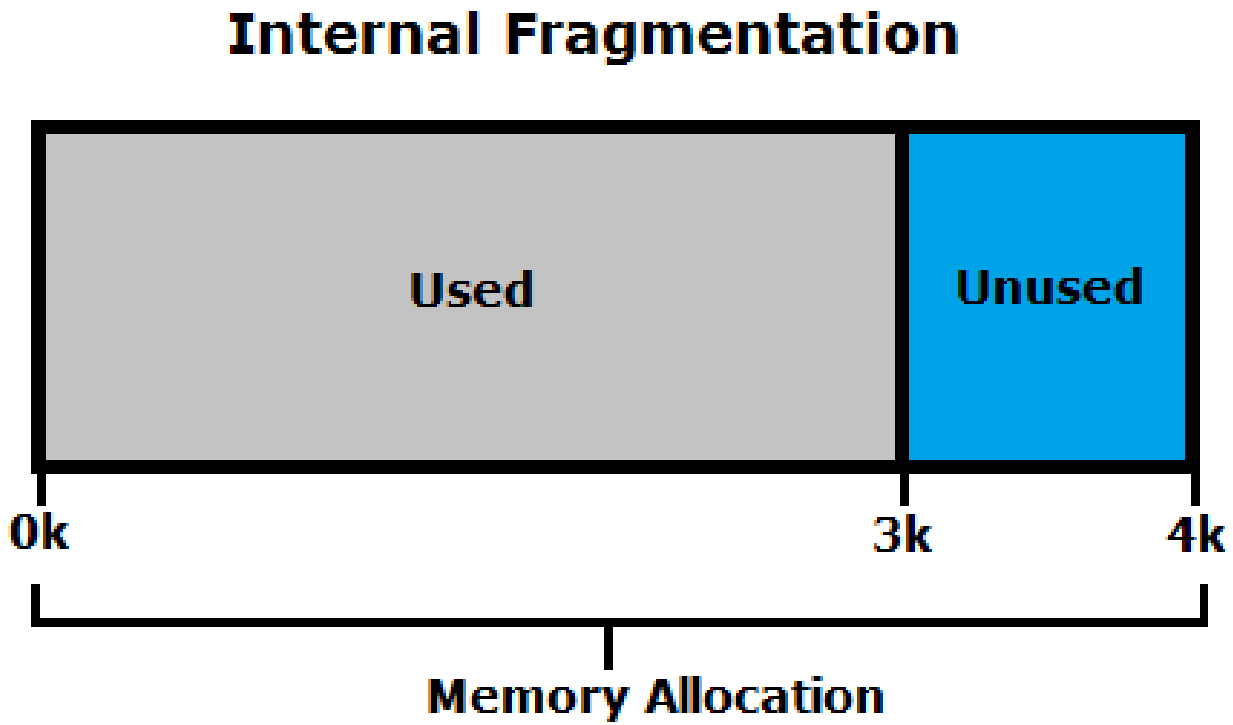
C++ style

- operator new;
- operator delete;

Performance Metrics

- Performance
  - CPU overhead
  - Memory overhead
- Scalability
- Memory Fragmentation
  - Internal Fragmentation
  - External Fragmentation
  - Memory Blowup
- Energy Consumption

Internal Fragmentation

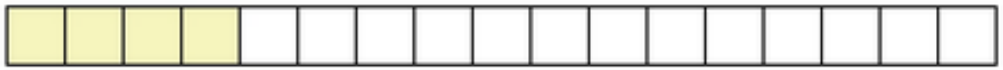


NOTE: 1k memory is wasted due to partitioning

External Fragmentation

- Occurs when there is enough aggregate heap memory, but no single free block is large enough

```
p1 = malloc(4)
```



```
p2 = malloc(5)
```

```
p3 = malloc(6)
```

```
free(p2)
```

```
p4 = malloc(6)
```

Oops! (what would happen now?)

- Depends on the pattern of future requests
  - Thus, difficult to measure

Blowup

- A special case of fragmentation:

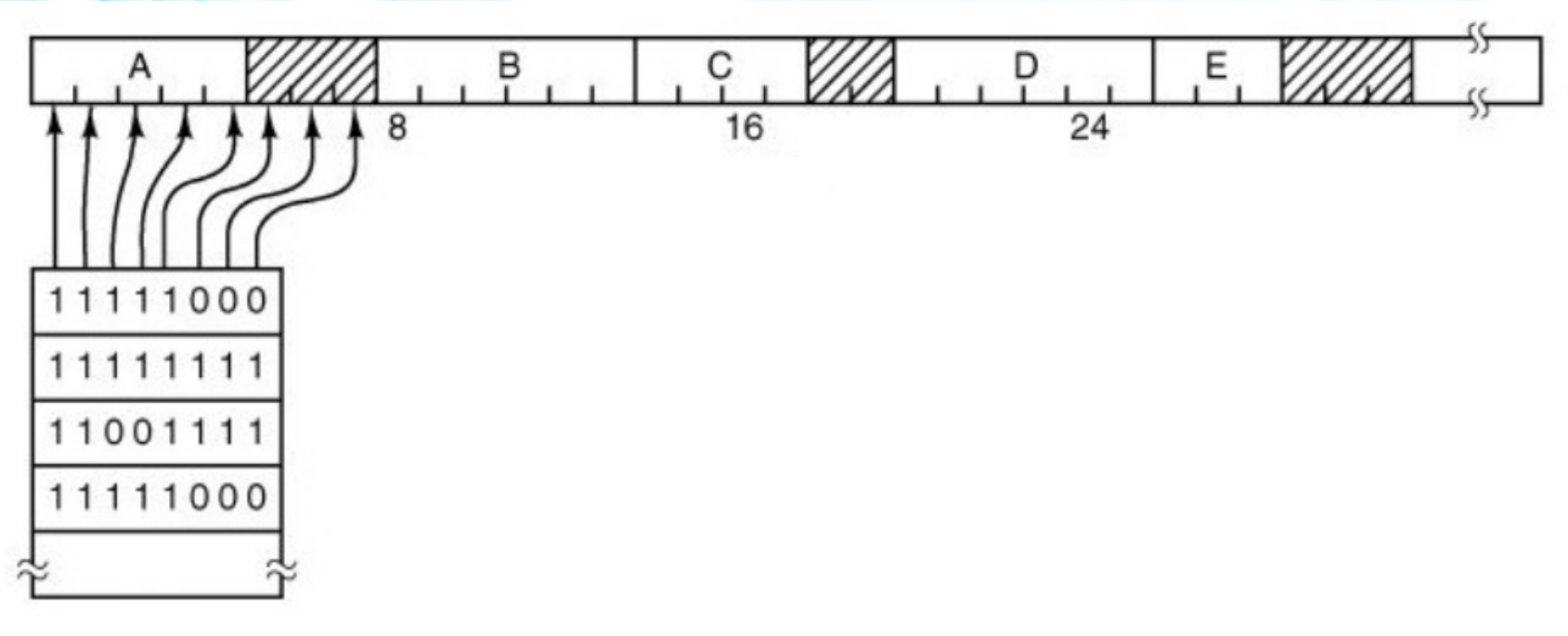
$$\frac{\texttt{max allocated}}{\texttt{max allocated by ideal uniprocessor allocator}}$$

- Unbounded, or grows linearly with # of CPUs
- Caused by parallel allocator not using freed memory to satisfy future allocation requests

Типы аллокаторов

- Bitmap-based
- List-based
- Tree-based
- Custom

Bitmap-based Allocators

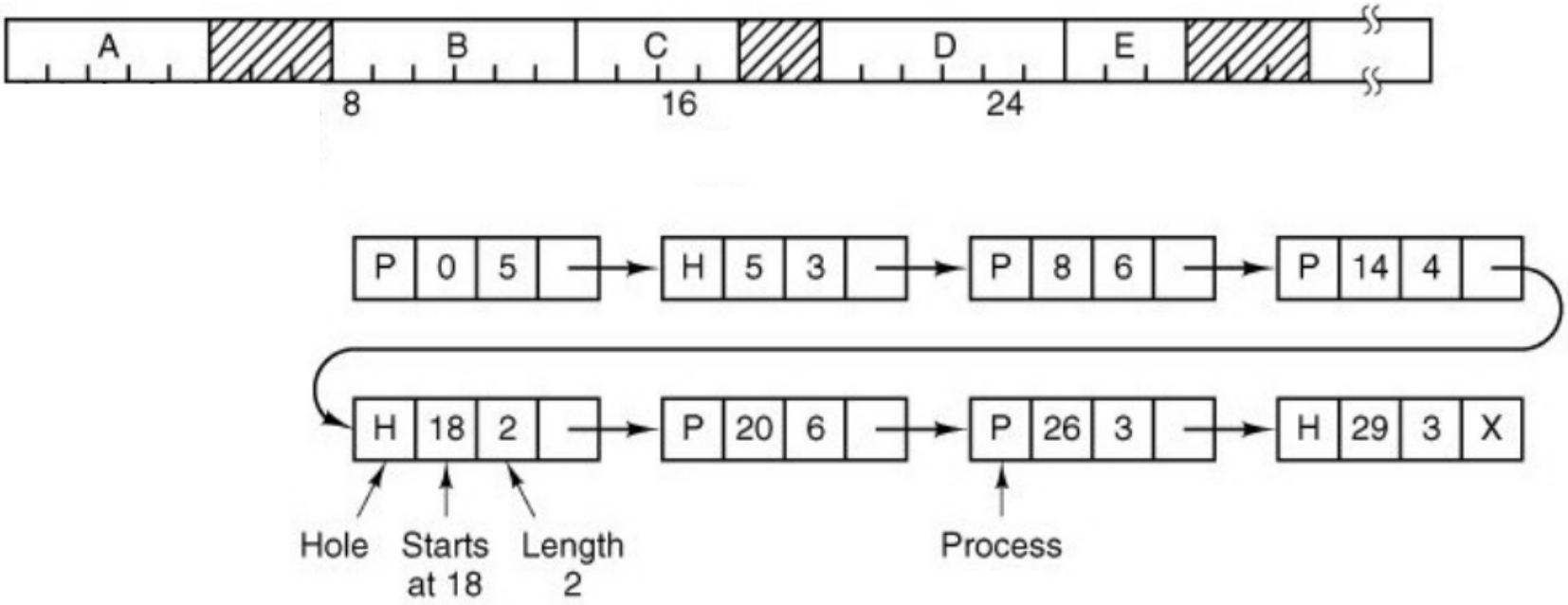


Bitmap-based Allocators

Основные особенности:

- Динамическая память и память необходимая для управления выделением разделены
- Линейная сложность выделения/освобождения
- Выделение на базе блоков
- Куча иммеет фиксированный размер
- Практически не требует взаимодействия с ОС
- Простая реализация
- Операции выделения и освобождения памяти являются дорогими

List-based Allocators

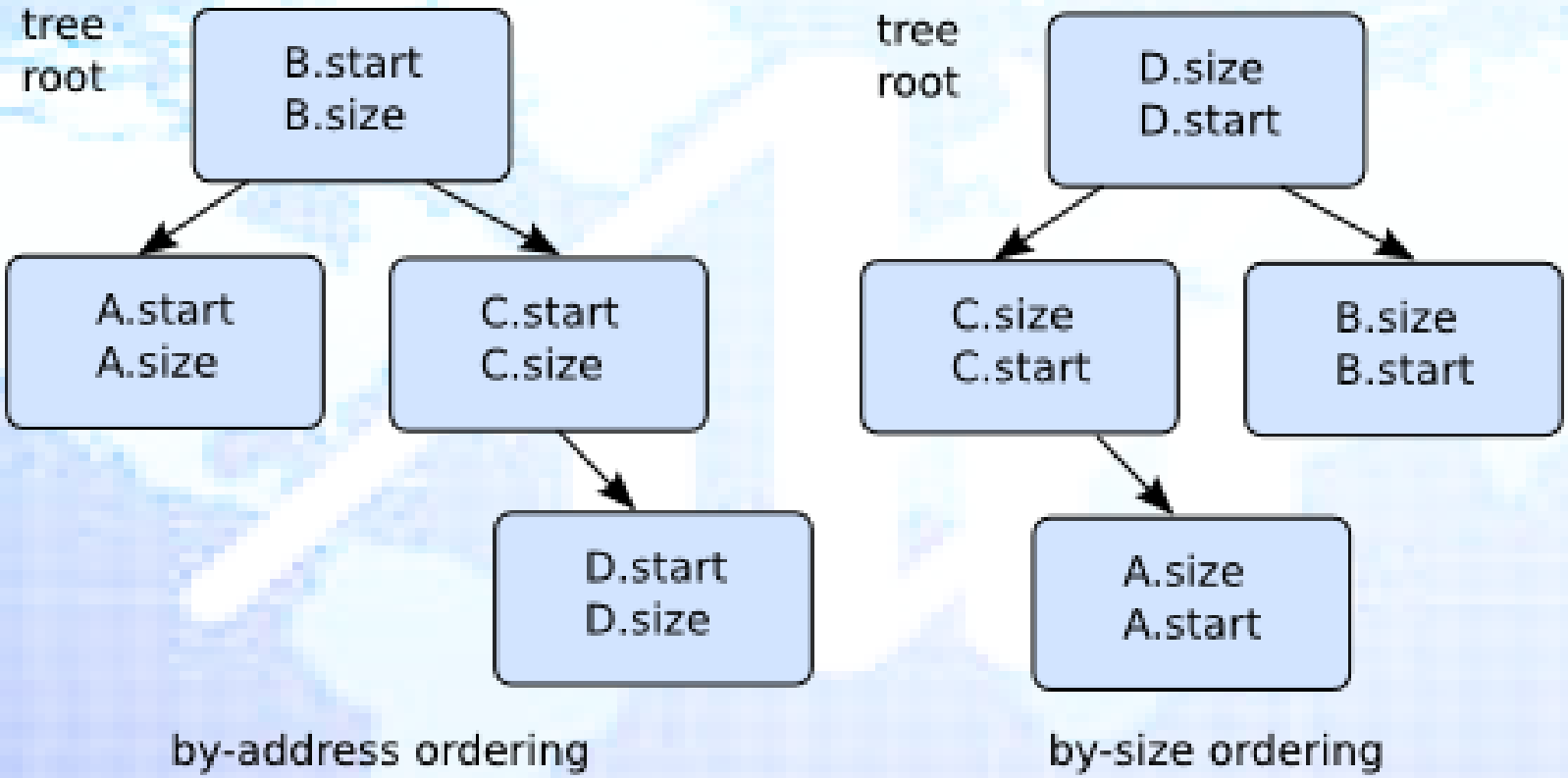


List-based Allocators

Основные особенности:

- Динамическая память используется как для обслуживания пользовательских запросов так и для внутренних нужд
- константная сложность выделения/освобождения
- Выделение на базе байтов
- Размер кучи может динамически меняться во время выполнения
- Активно взаимодействует с ОС
- Простая реализация
- Операции выделения и освобождения памяти являются дешевыми

Tree-based Allocators



Tree-based Allocators

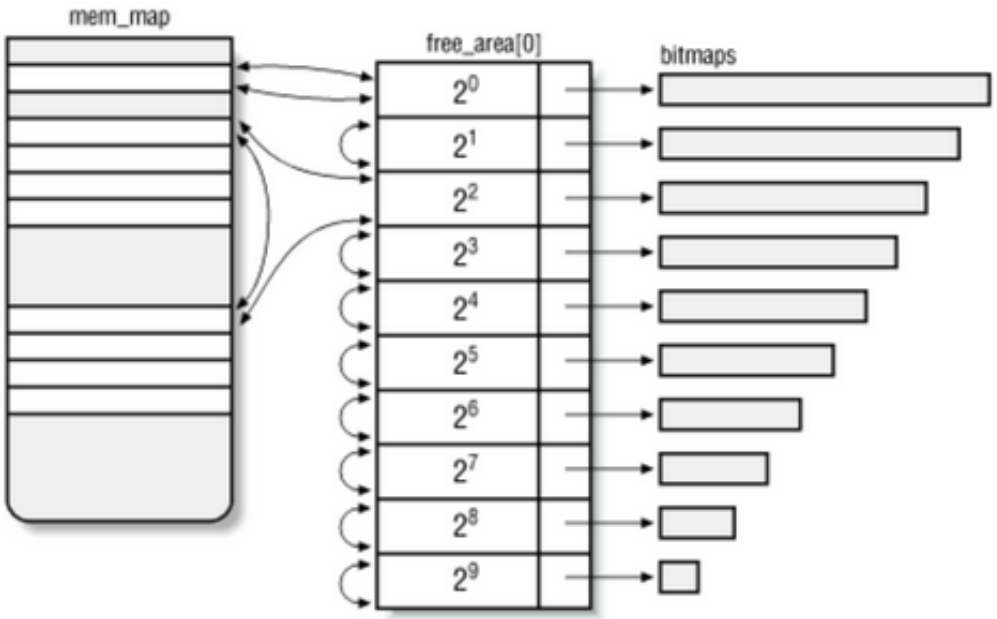
Основные особенности:

- Динамическая память используется как для обслуживания пользовательских запросов так и для внутренних нужд
- Логарифмическая сложность выделения/освобождения
- Выделение на базе байтов
- Размер кучи может динамически меняться во время выполнения
- Активно взаимодействует с ОС
- Простая реализация
- Операции выделения и освобождения памяти являются дешевыми
- Применение политик выделения памяти является дешевым

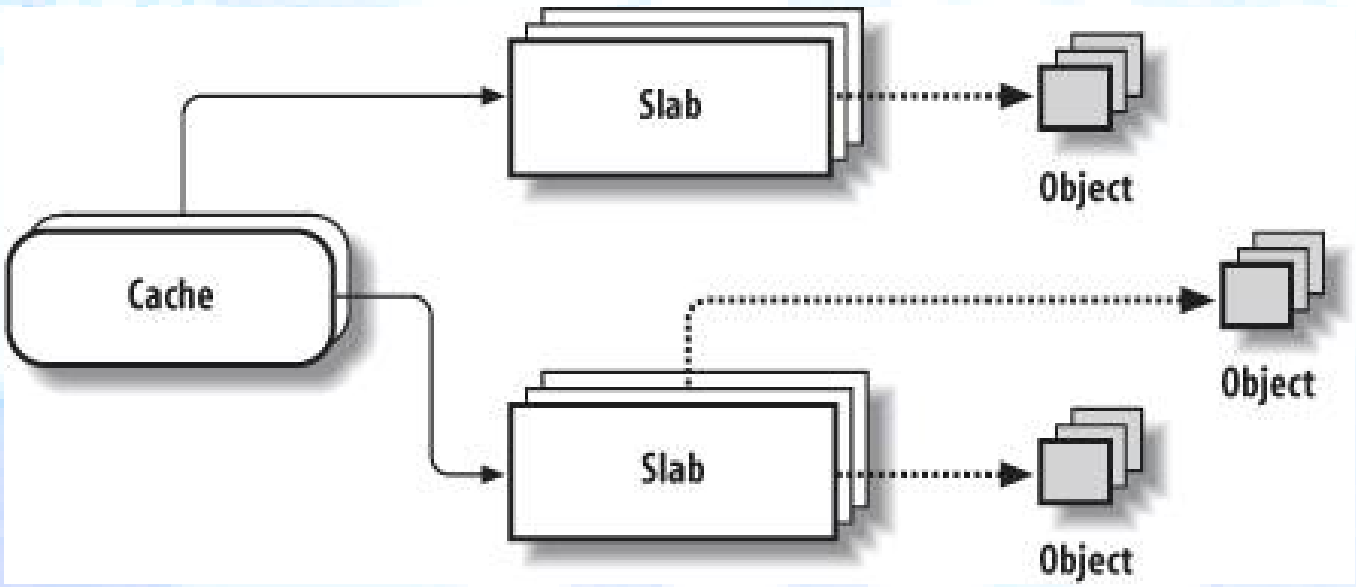
Политики выделения памяти

- First Fit
- Best Fit
- Worst Fit

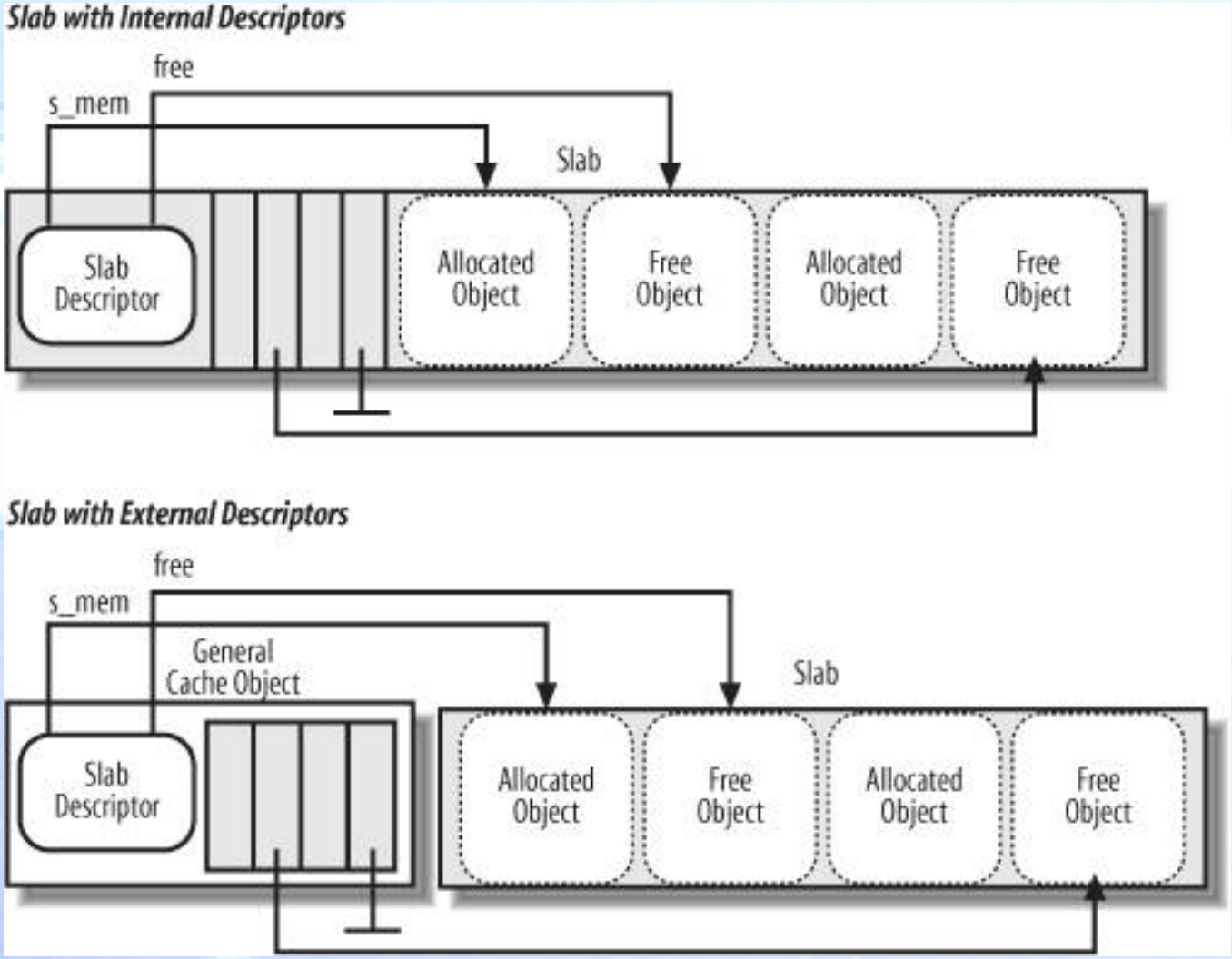
Buddy Allocator



Slab Allocator



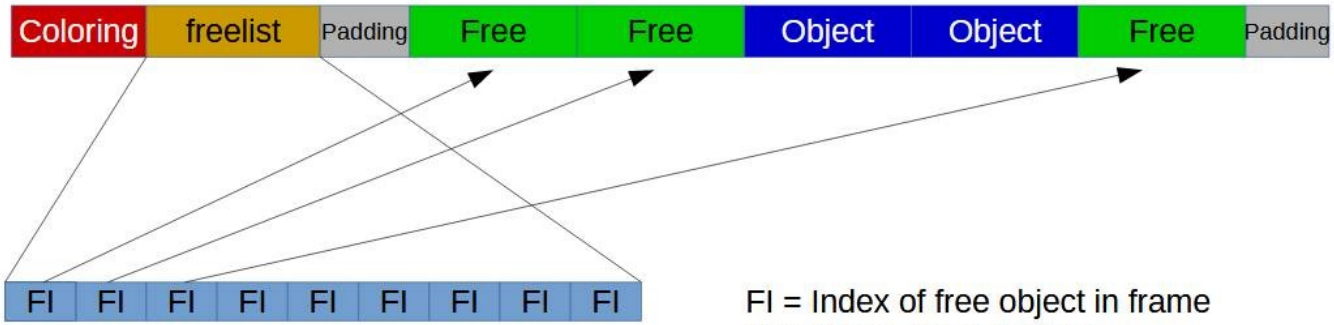
Slab Allocator



Slab Allocator

SLAB per frame freelist management

Page Frame Content:



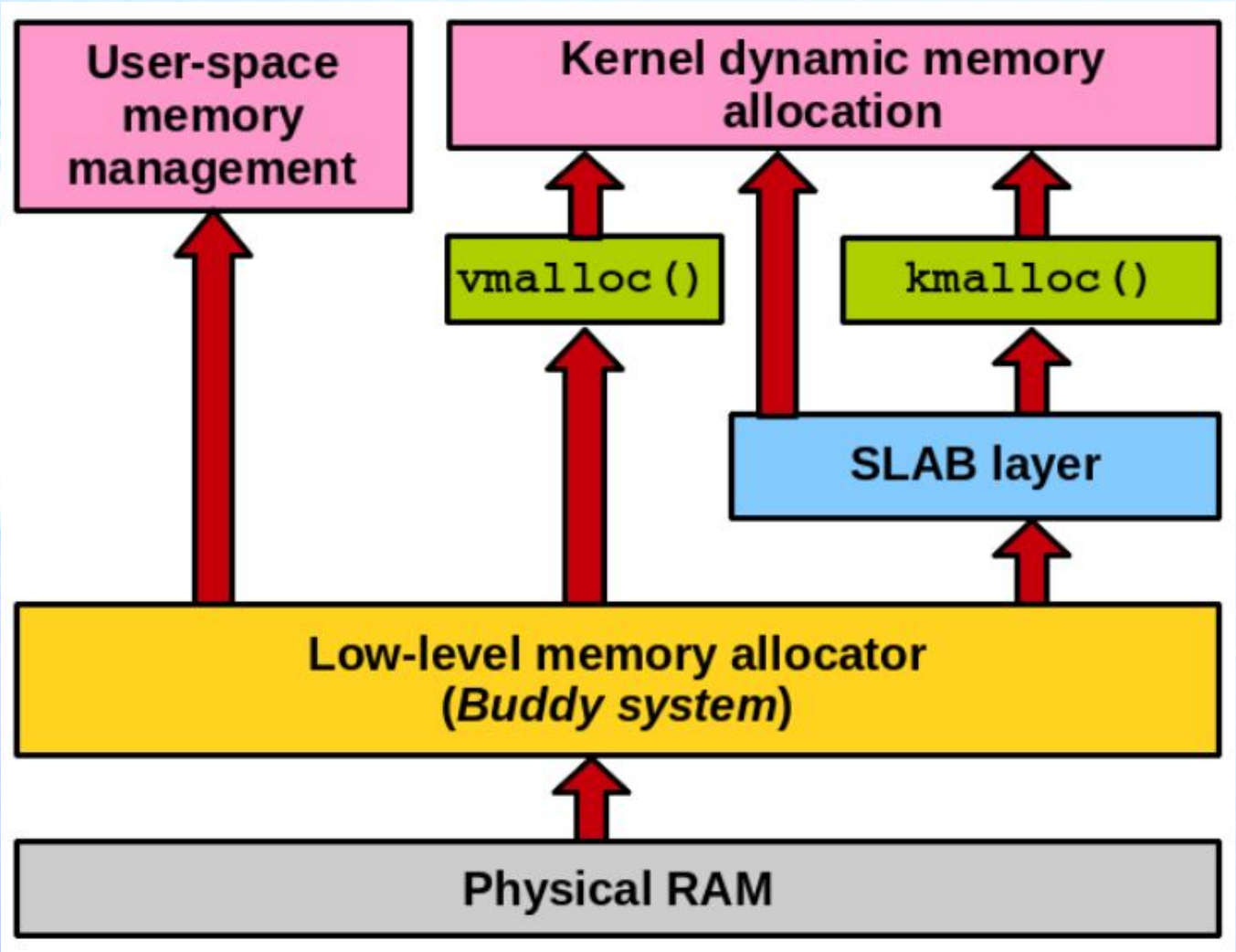
FI = Index of free object in frame
Two types: short or char

Page->active

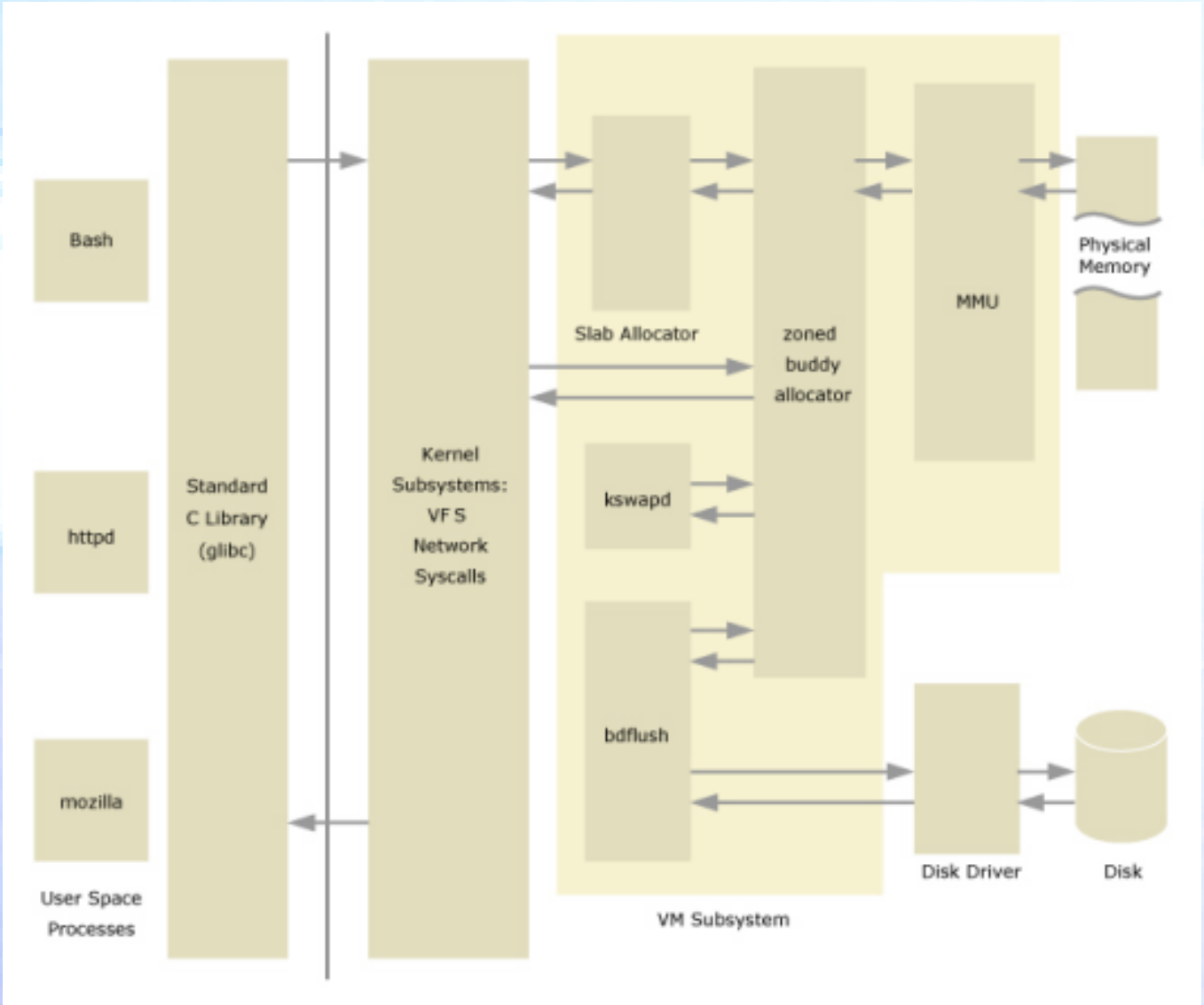
For each object in the frame

Multiple requests for free objects can be satisfied from the same cacheline without touching the object contents.

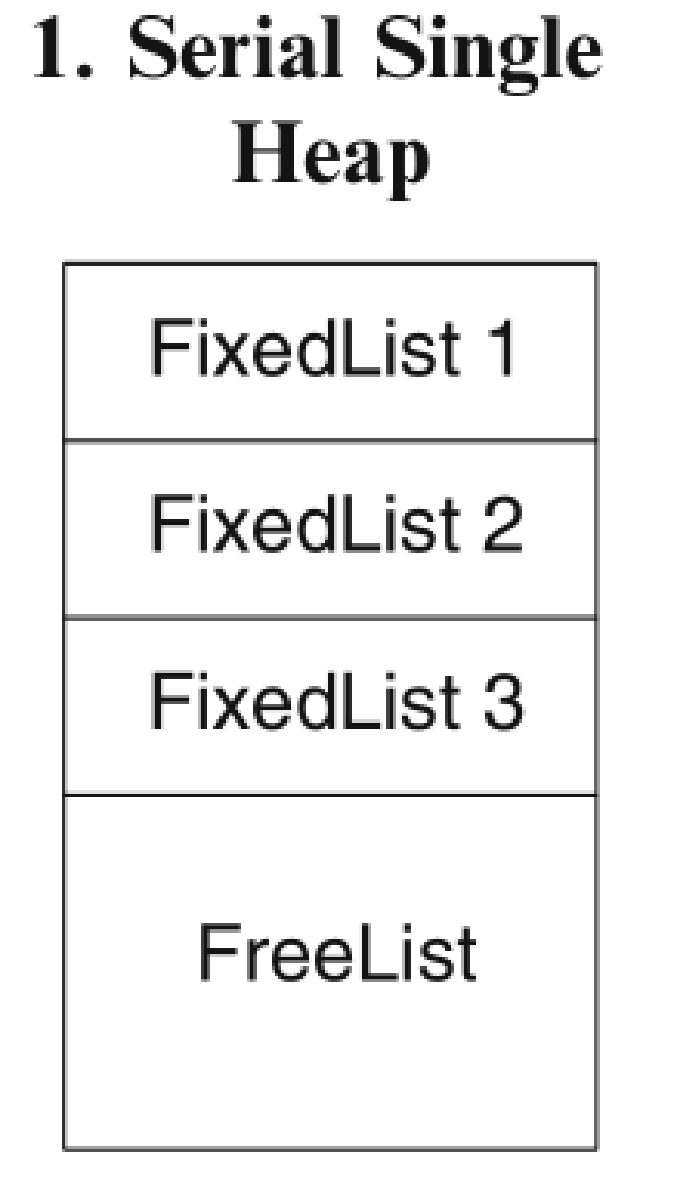
Low-level memory allocator (Linux)

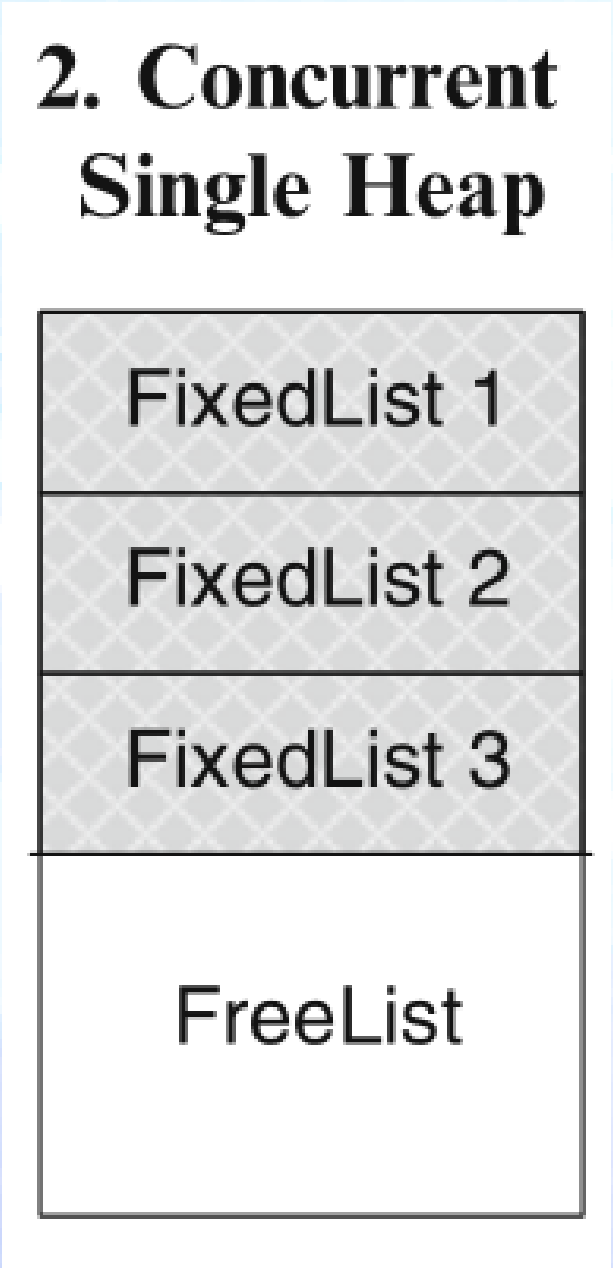


High overview of Virtual Memory subsystem (Linux)

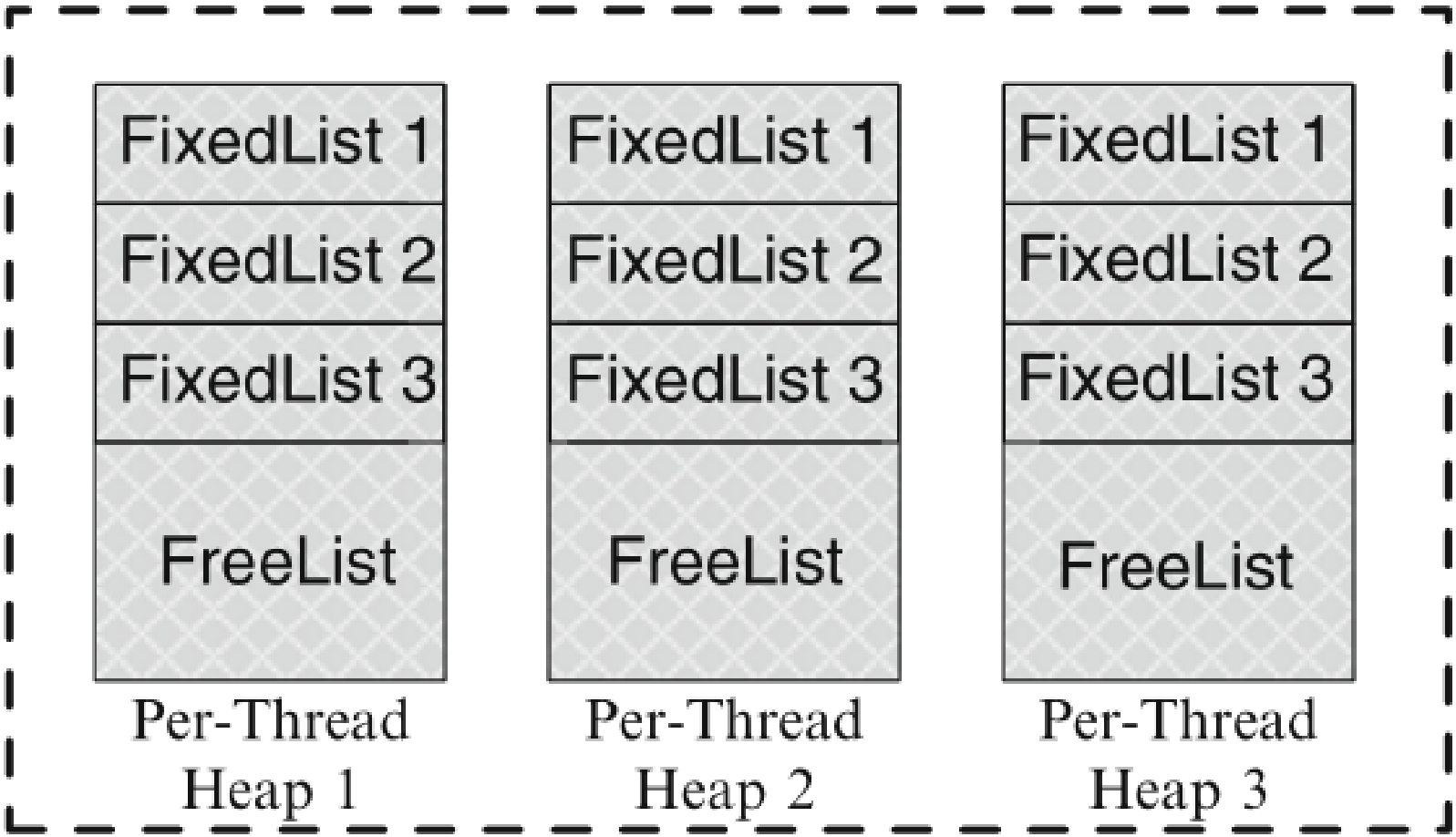


Архитектура кучи

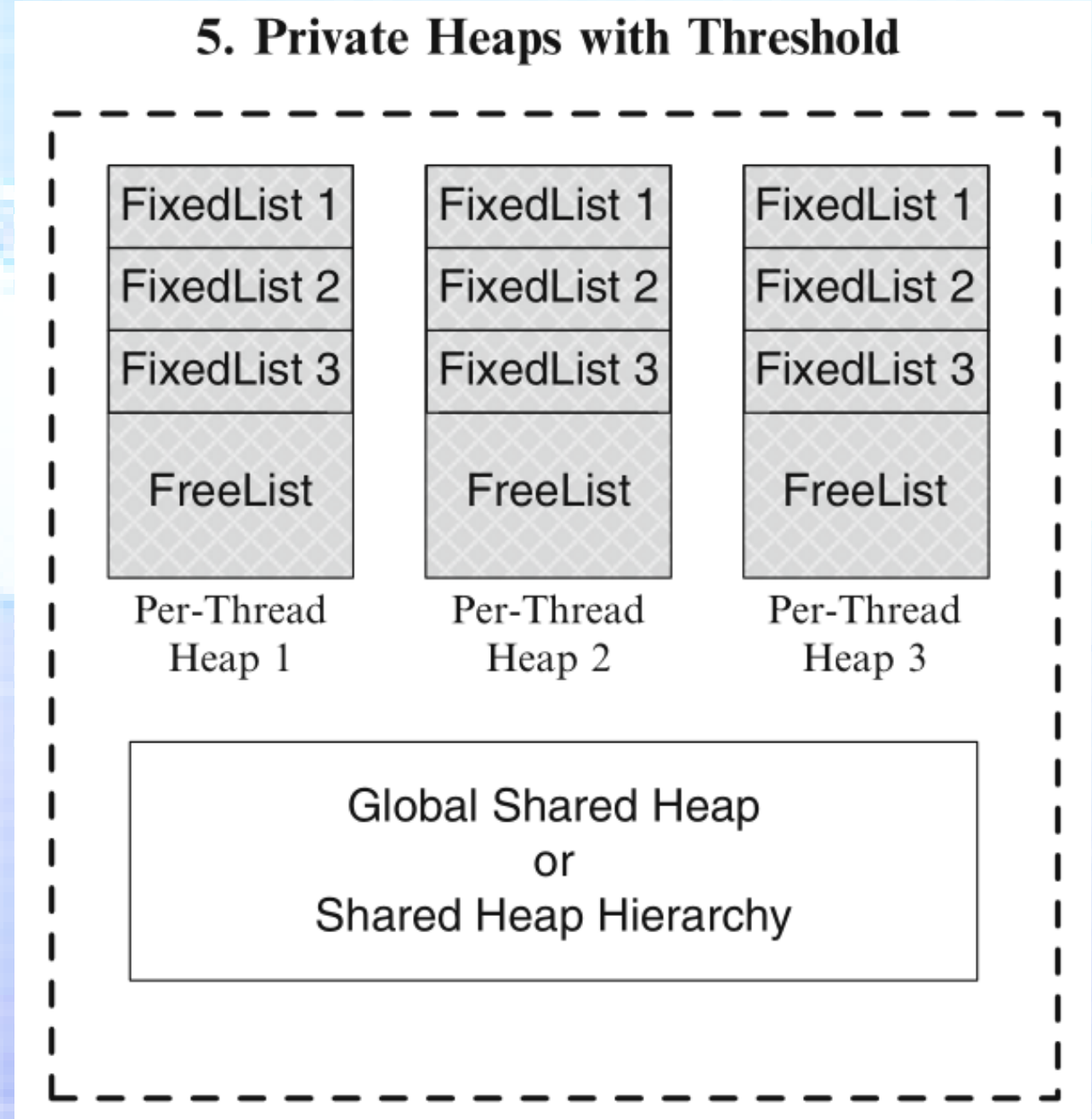



2. Concurrent Single Heap
FixedList 1
FixedList 2
FixedList 3
FreeList

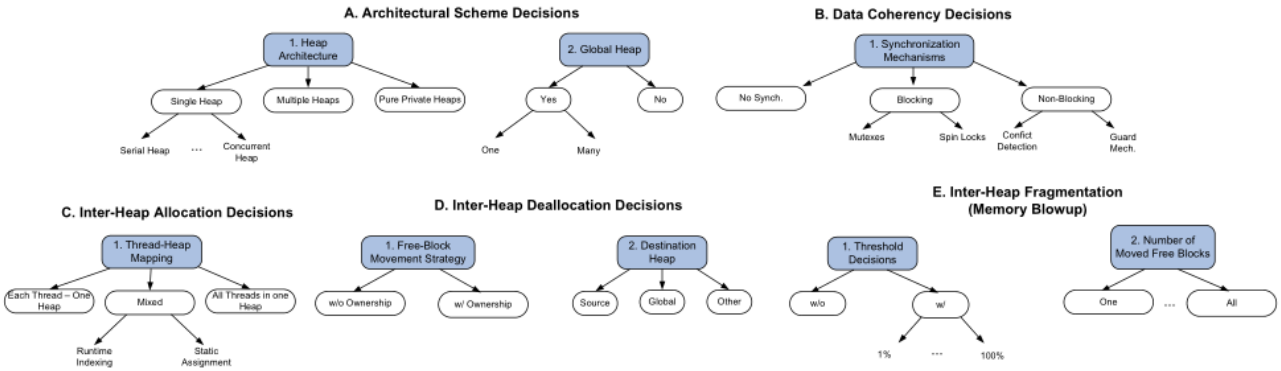
3. Pure Private Heaps and
4. Private Heap with Ownership



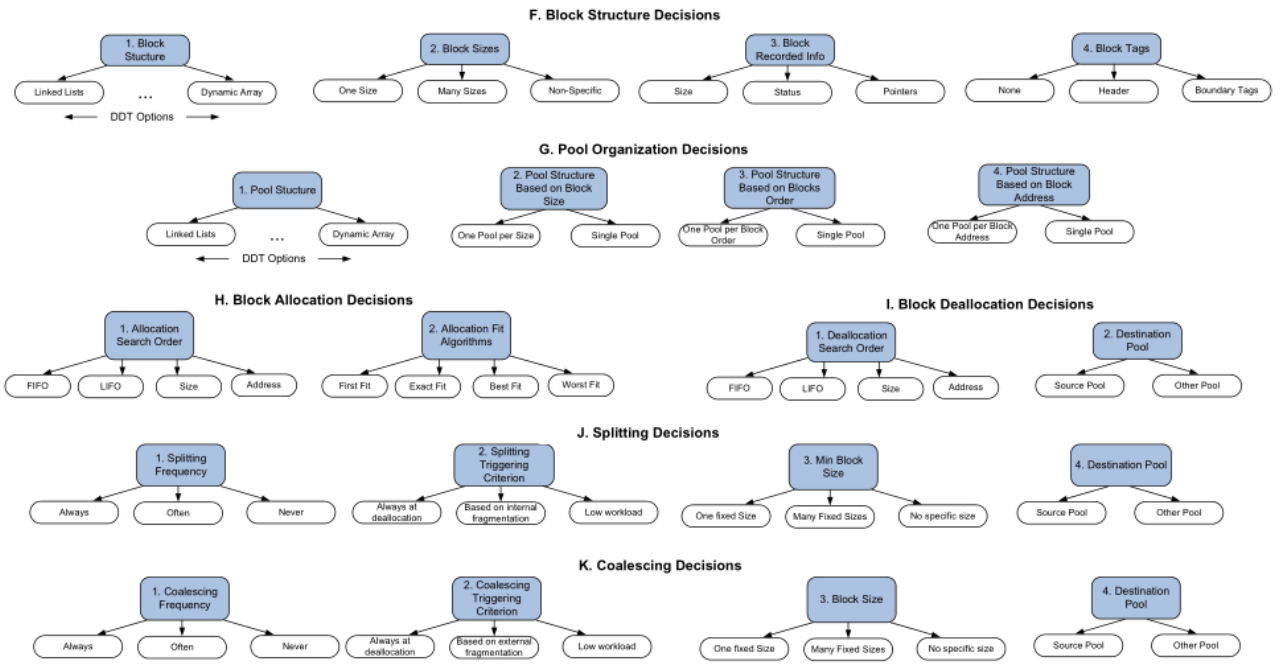
Архитектура кучи



Design Space

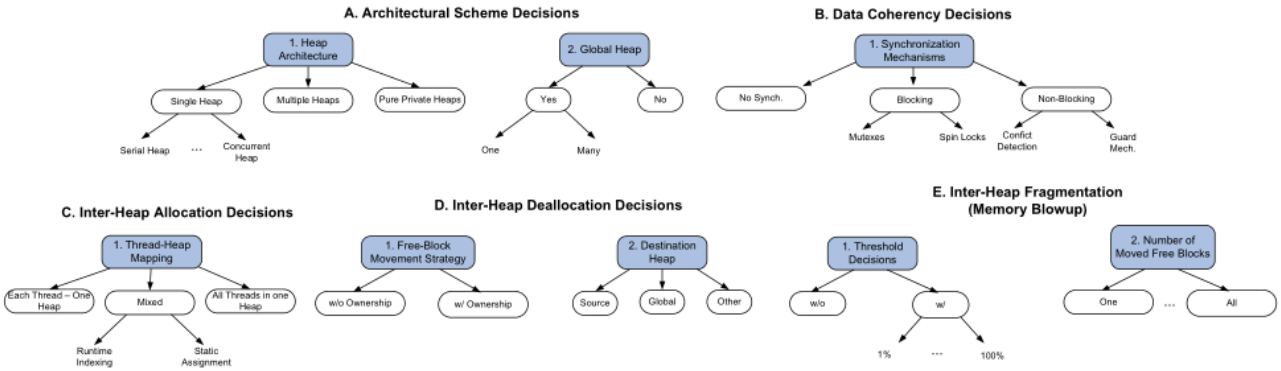


(a) Inter-heap level design space.

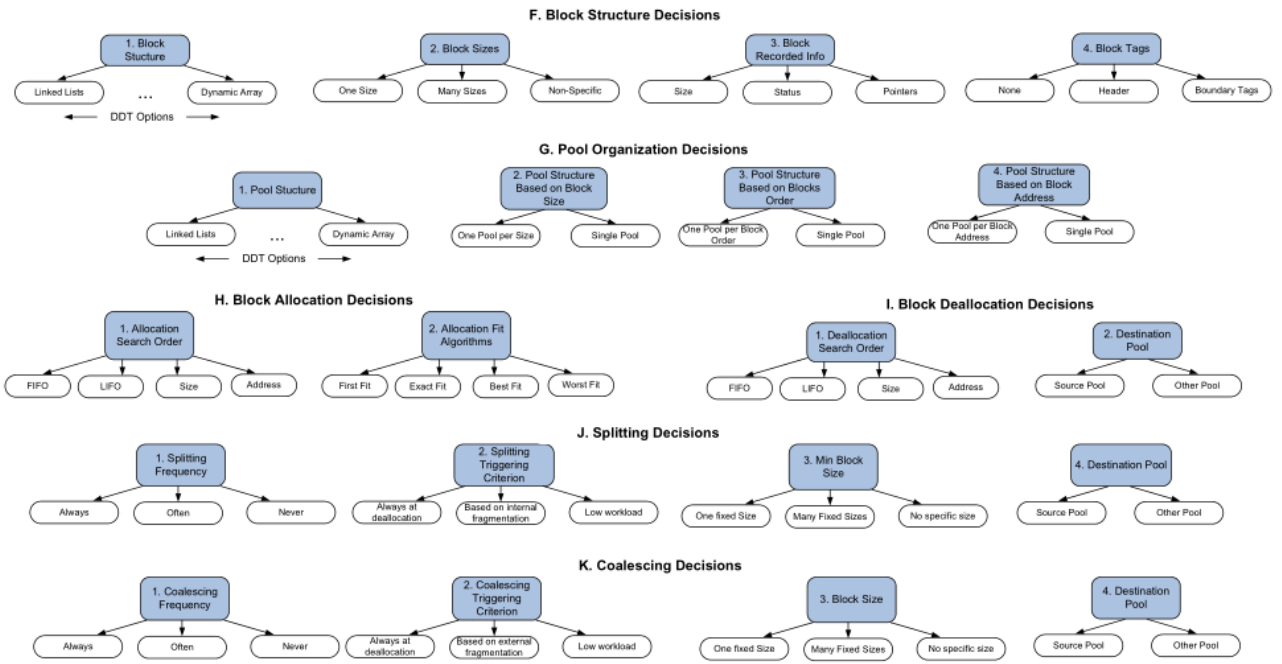


(b) Intra-heap level design space.

Design Space

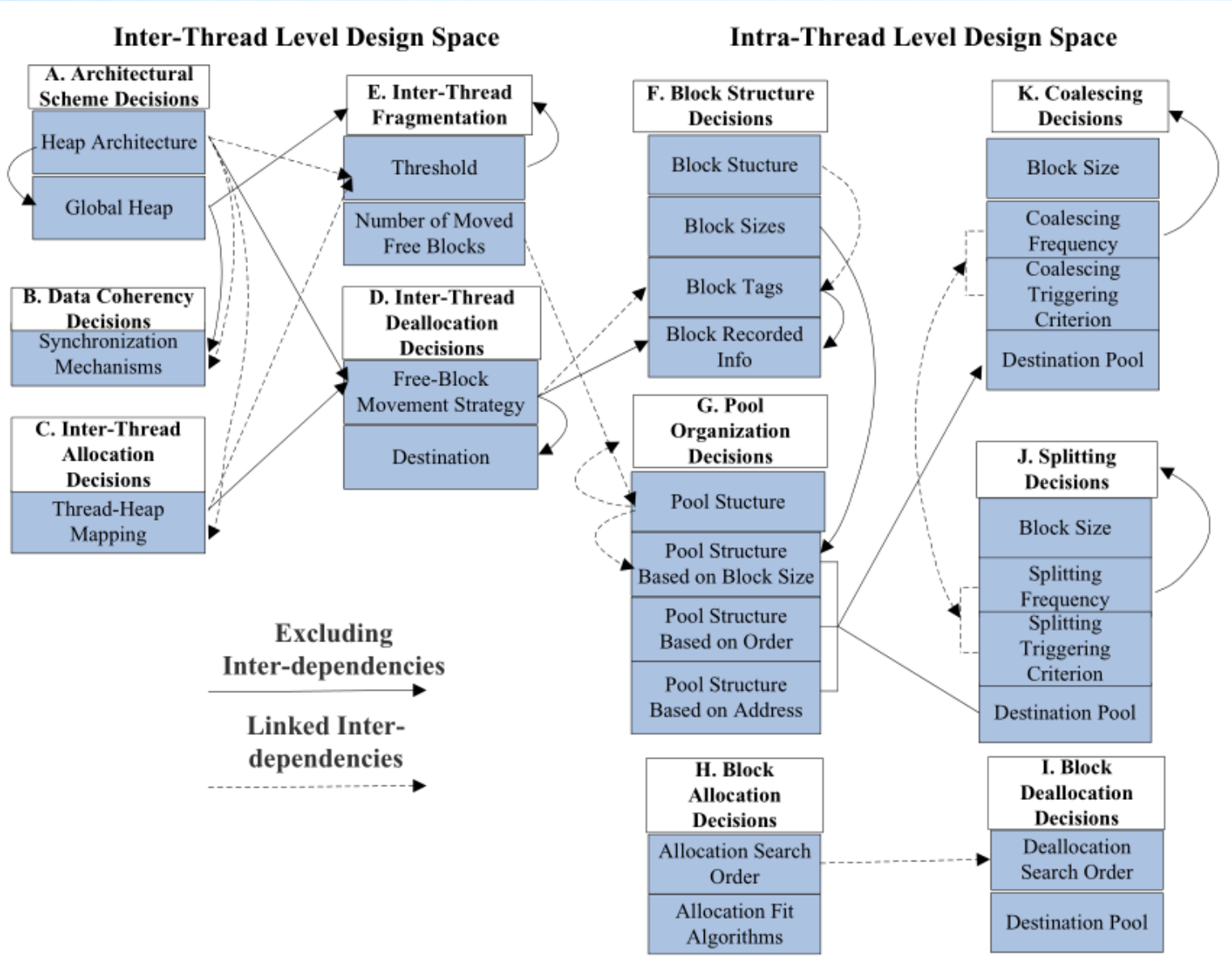


(a) Inter-heap level design space.

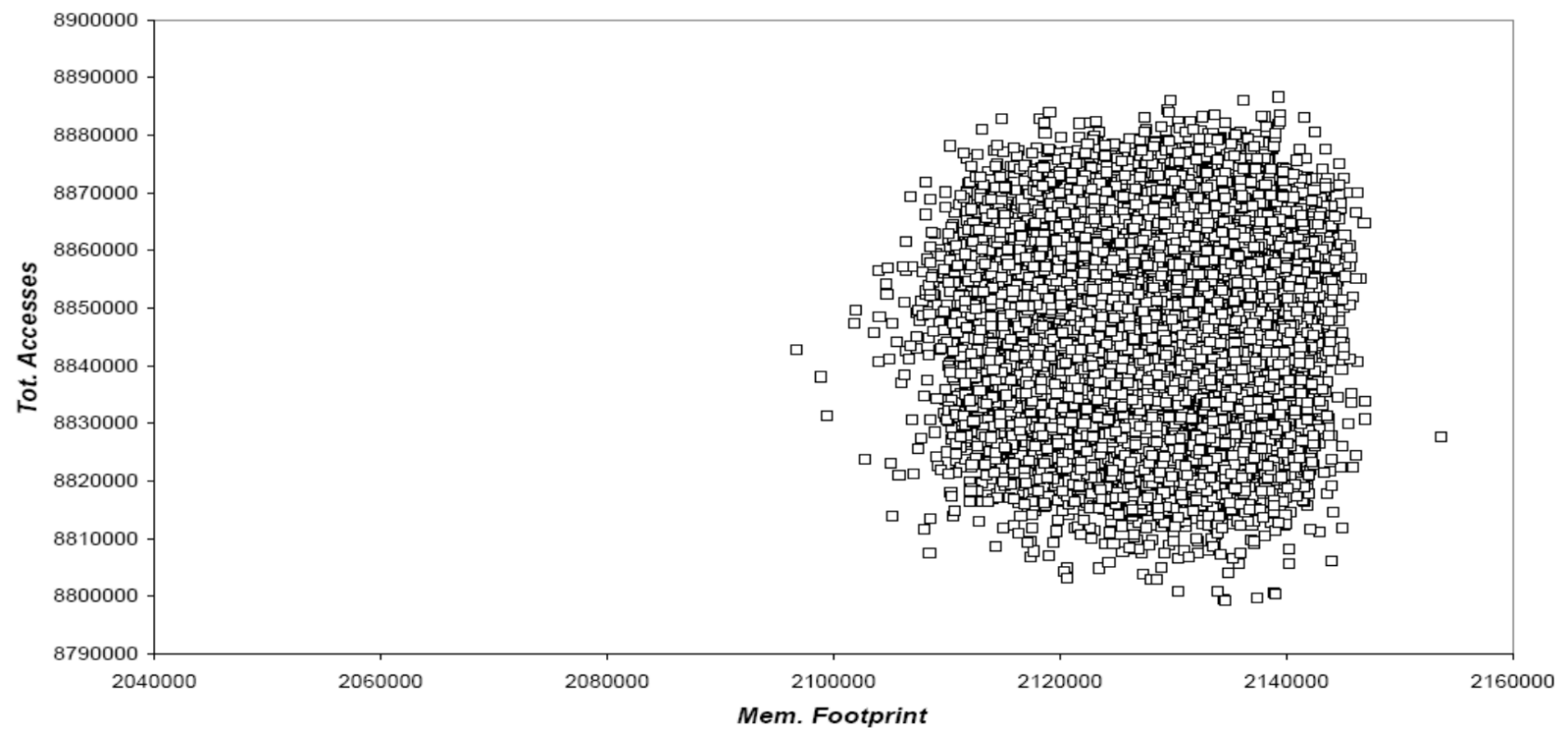


(b) Intra-heap level design space.

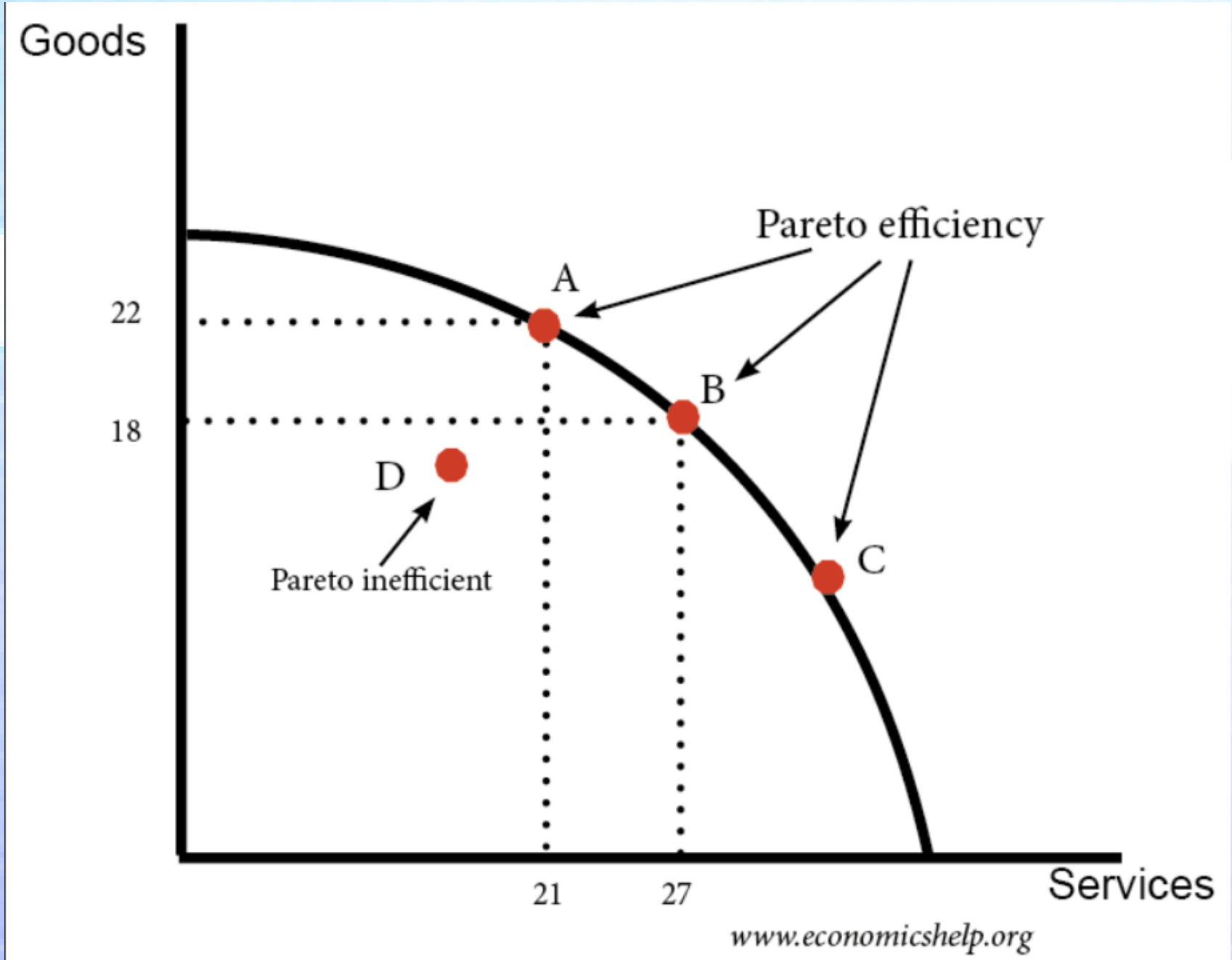
Design Space



Design Space



Оптимальность по Парето



Оптимальность по Парето