

## 8. 高级搜索树

### (b1) B-树：动机

640K ought to be enough for anybody.

- B. Gates, 1981

邓俊辉

deng@tsinghua.edu.cn

## 越来越小的内存

❖ RAM : 存储器？不就是无限可数个寄存器吗？

Turing : 存储器？不就是无限长的纸带吗？

❖ 但事实上

系统存储容量的增长速度

<< 应用问题规模的增长速度

1 Kilobyte =  $2^{10} = 10^3$

1 Megabyte =  $2^{20} = 10^6$

1 Gigabyte =  $2^{30} = 10^9$

1 Terabyte =  $2^{40} = 10^{12}$

1 Petabyte =  $2^{50} = 10^{15}$

1 Exabyte =  $2^{60} = 10^{18}$

1 Zettabyte =  $2^{70} = 10^{21}$

1 Yottabyte =  $2^{80} = 10^{24}$

1 Nonabyte =  $2^{90} = 10^{27}$

1 Doggabyte =  $2^{100} = 10^{30}$

2010

## 越来越小的内存

❖ 典型的

1980 : 10MB / 1MB = 10

数据库规模 / 内存容量

2000 : 1TB / 1GB = 1000

❖ 今天典型的数据集

345 TB ^ Global climate

须以TB为单位度量

300 TB ^ Nuclear

250 TB ^ Turbulent combustion

50 TB ^ Parkinson's disease

10 TB ^ Protein folding

❖ 亦即，相对而言...内存容量是在...不断减小！

❖ 为什么不把内存做得更大？

❖ 物理上，存储器的容量越大/小，访问速度就越慢/快

## 高速缓存

❖ 事实1：不同容量<sub>的存储器</sub>，访问速度<sub>差异悬殊</sub>

❖ 以磁盘与内存为例： $\text{ms} / \text{ns} > 10^5$

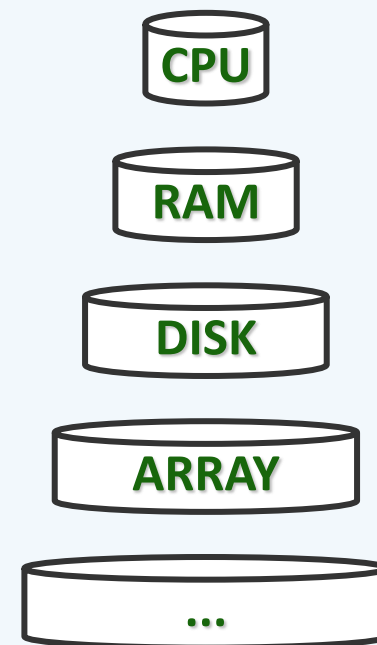
❖ 若一次内存访问需要一秒，则一次外存访问就相当于一天

❖ 为避免1次外存访问，我们宁愿访问内存10次、100次，甚至...

❖ 多数存储系统，都是分级组织的——Caching  
最常用的数据尽可能放在更高层、更小的存储器中  
实在找不到，才向更低层、更大的存储器索取

❖ 算法的I/O复杂度  $\propto$  数据在不同存储级别之间的传输次数

算法的实际运行时间，往往主要取决于此



## 高速缓存

❖ 事实2：从磁盘中读写1B，与读写1KB几乎一样快

❖ 批量式访问：以页（page）或块（block）为单位，使用缓冲区 //<stdio.h>...

❖



```
#define BUFSIZ 512 //缓冲区默认容量
```

```
int setvbuf( //定制缓冲区
```

```
FILE* fp, //流
```

```
char* buf, //缓冲区
```

```
int _Mode, //_IOFBF | _IOLBF | _IONBF
```

```
size_t size); //缓冲区容量
```

```
int fflush(FILE* fp); //强制清空缓冲区
```

❖ 效果：单位字节的1KB访问时间大大缩短