# Performance Improvement of OpenLDAP Transactional Backend

Jong Hyuk Choi
jongchoi@us.ibm.com
IBM Thomas J. Watson Research Center
Enterprise Linux Group

Mar 21, 2003

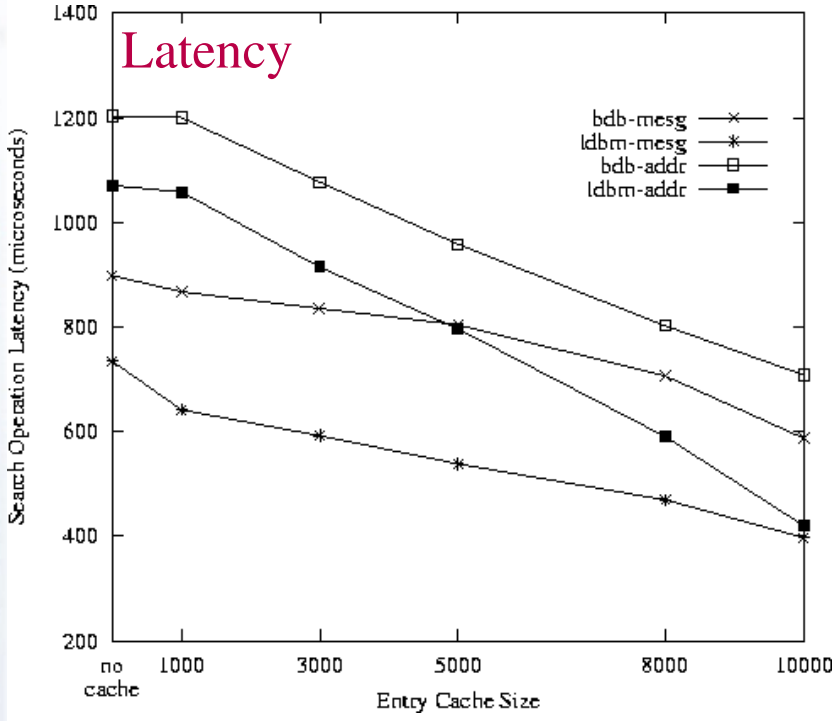
# OpenLDAP Transactional Backend

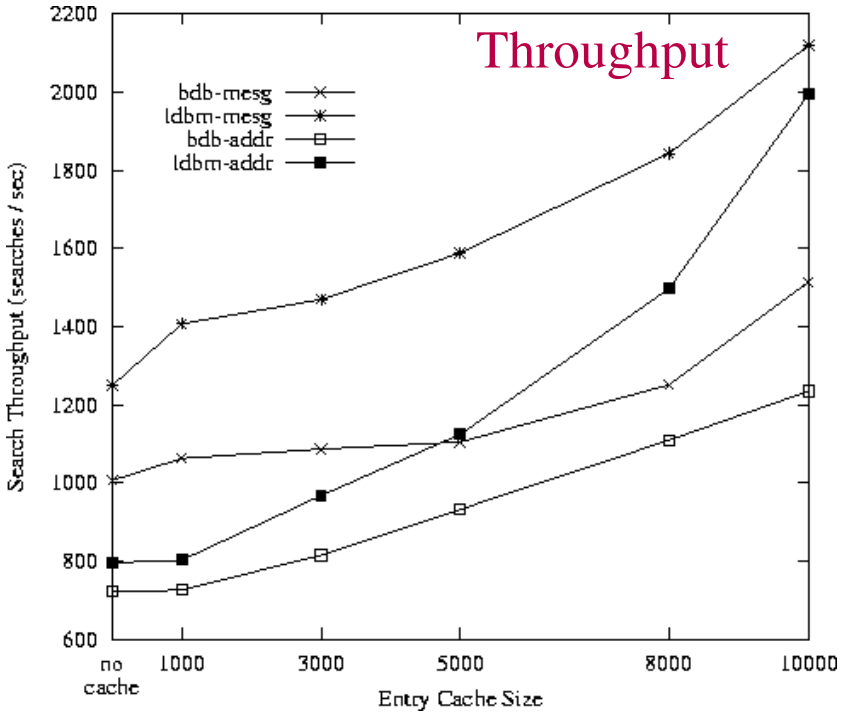
## Transactional Backend : back-bdb

- A New backend of recently released OpenLDAP 2.1
  - Use native Berkeley DB API - BDB transactional store
  - Transaction-protected updates via write-ahead logging
    - Rollback upon transient errors such as deadlock
    - Recovery from catastrophic failures
  - Page-level locking
    - Back-ldbm : backend giant lock
    - Improved concurrency
    - Concurrent execution of slapd and admin tools (slapcat, slapadd...)
  - Improvements
    - Store DB data in a binary format : reduce mallocs
    - Simple IDL management
- Back-bdb performance
  - Overhead of BDB transaction env for non-transaction ops

# Back-bdb Entry Cache

- **Entry cache**
  - Cache for the id2entry database
  - Essential for high performance directory searches
  - Evolved from back-ldbm entry cache
- **Locking Issues in back-bdb**
  - Entry level locking interferes with BDB page-level locking
  - To avoid deadlock, entry cache locking was redesigned to use BDB locking primitives

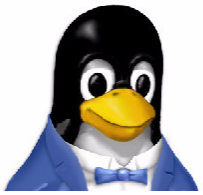


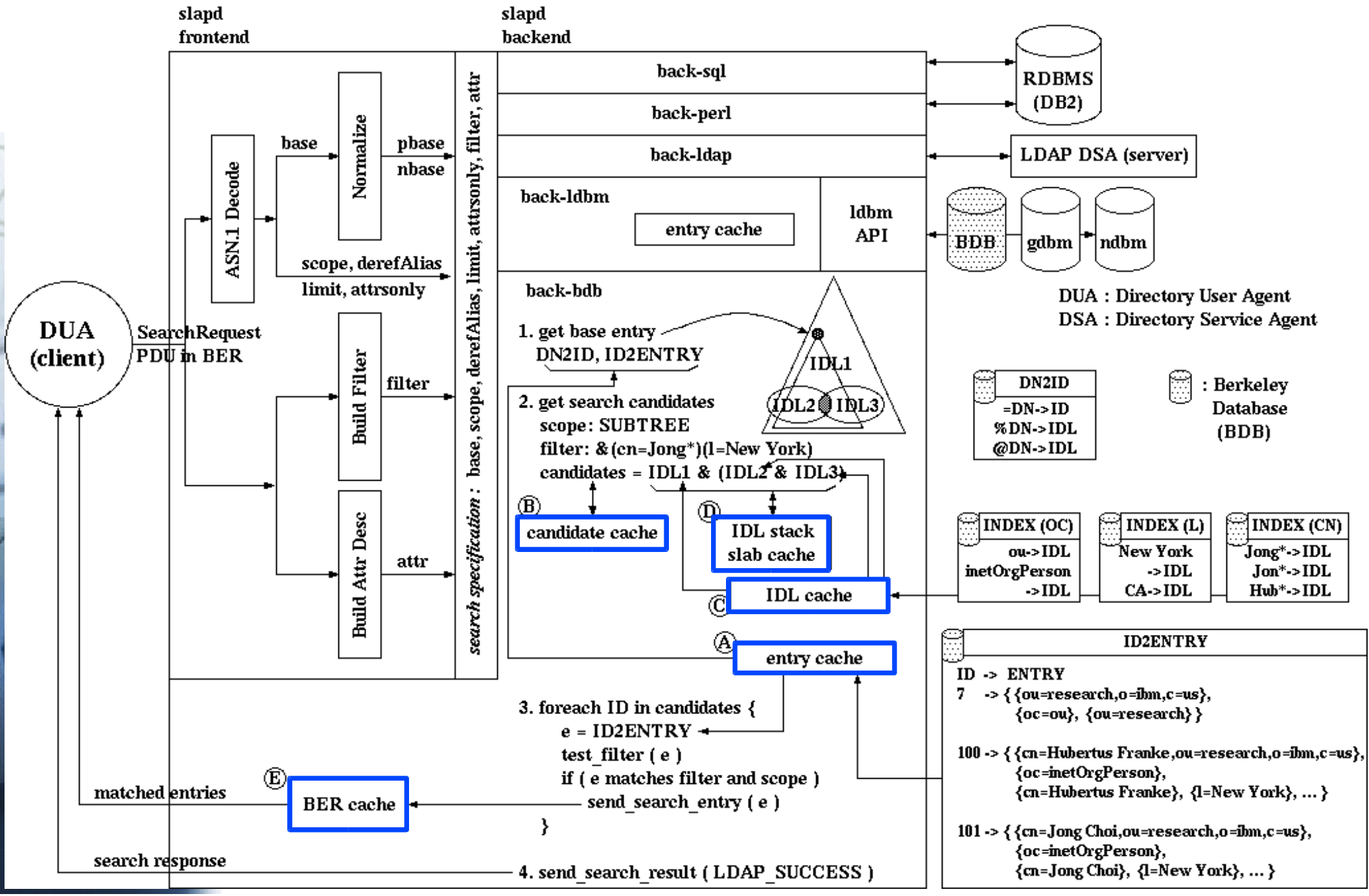
# Transaction Overhead

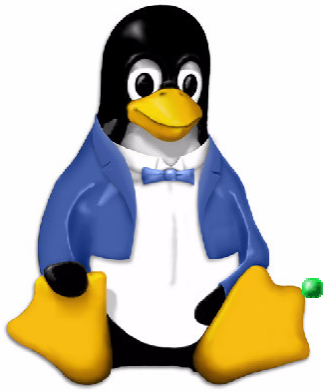
## Transaction Overhead

- Hash back-ldbm w/o transaction environment
  - DirMark (mesg) : 1956.8 ops/sec
- Hash back-ldbm w transaction environment
  - `DB_INIT_TXN | DB_INIT_LOG | DB_INIT_LOCK | DB_INIT_MPOOL`
  - DirMark (mesg) : 1821.9 ops/sec (down 6.9%)
- Btree back-ldbm w/o transaction environment
  - DirMark (mesg) : 2006.6 ops/sec
- Btree back-ldbm w transaction environment
  - `DB_INIT_TXN | DB_INIT_LOG | DB_INIT_LOCK | DB_INIT_MPOOL`
  - DirMark (mesg) : 1932.3 ops/sec (down 3.7%)
- Transaction overhead exists for non transaction-protected operations



# Dissection of Search Operation





# Base System Profiling

## IBM Trace Facility for Linux (x86)

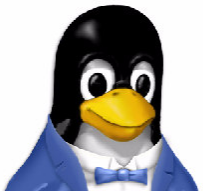
- Use Pentium performance counter
- INST_RETIRED event with x10000 sampling

### Back-bdb

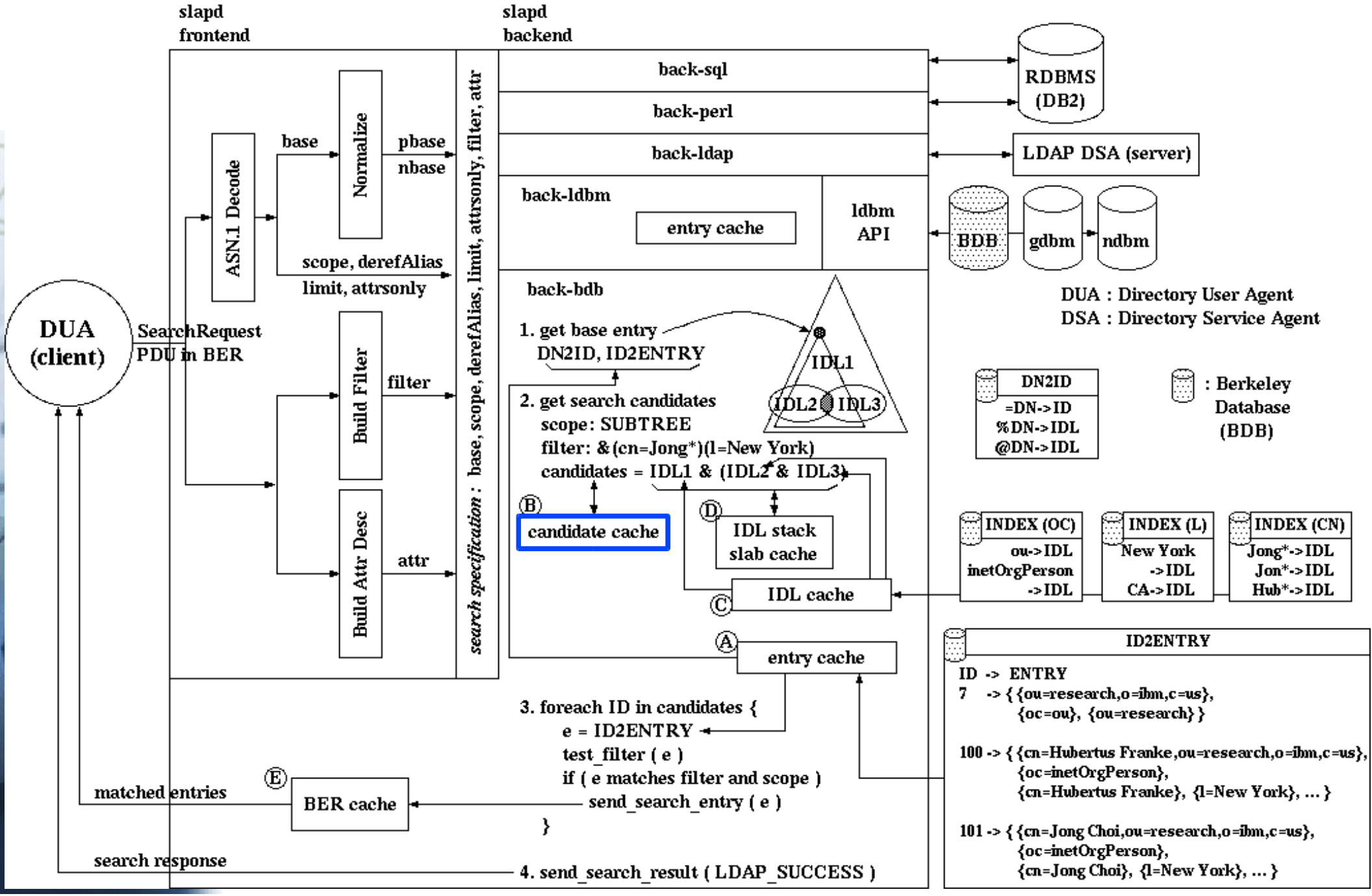
| | | | | | |
|---|---|---|---|---|---|
| **slapd** | 31.47% | **libdb-4.0** | 26.99% | **vmlinux** | 13.79% |
| ber_printf | 2.01% | _ham_item | 6.55% | do_zap_page_range | 2.26% |
| send_search_entry | 1.81% | _ham_lookup | 6.54% | get_unmapped_area | 0.72% |
| ber_write | 1.51% | _ham_get_cpage | 2.22% | save_i387 | 0.60% |
| is_ad_subtype | 1.48% | _ham_item_next | 1.81% | do_signal | 0.39% |
| ad_inlist | 1.29% | _lock_get_internal | 1.16% | schedule | 0.38% |
| avl_find | 1.26% | _db_tas_mutex_lock | 0.91% | tcp_sendmsg | 0.37% |
| **libc** | 15.57% | **libpthread** | 9.09% | **misc** | 3.09% |

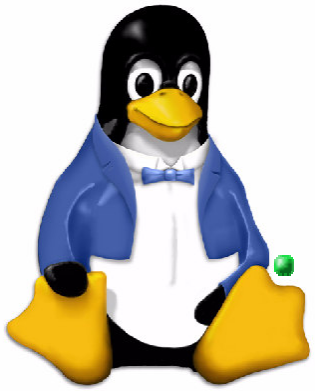
### Back-ldbm

| | | | | | |
|---|---|---|---|---|---|
| **slapd** | 42.72% | **libdb-4.0** | 4.44% | **vmlinux** | 11.44% |
| ber_printf | 2.58% | _bam_cmp | 0.87% | save_i387 | 0.77% |
| send_search_entry | 2.46% | _bam_search | 0.61% | schedule | 0.70% |
| ber_write | 2.05% | _memp_fget | 0.40% | send_sig_info | 0.56% |
| is_ad_subtype | 1.99% | _db_c_get | 0.29% | restore_i387 | 0.54% |
| ber_put_seqorset | 1.74% | _memp_fput | 0.27% | tcp_sendmsg | 0.53% |
| ad_inlist | 1.53% | _db_icursor | 0.25% | do_signal | 0.47% |
| **libc** | 23.08% | **libpthread** | 14.06% | **misc** | 4.26% |



# Candidate Caching





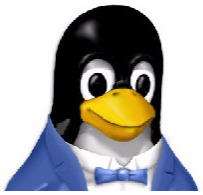
# Candidate Caching

## Caching IDL of Search Candidates

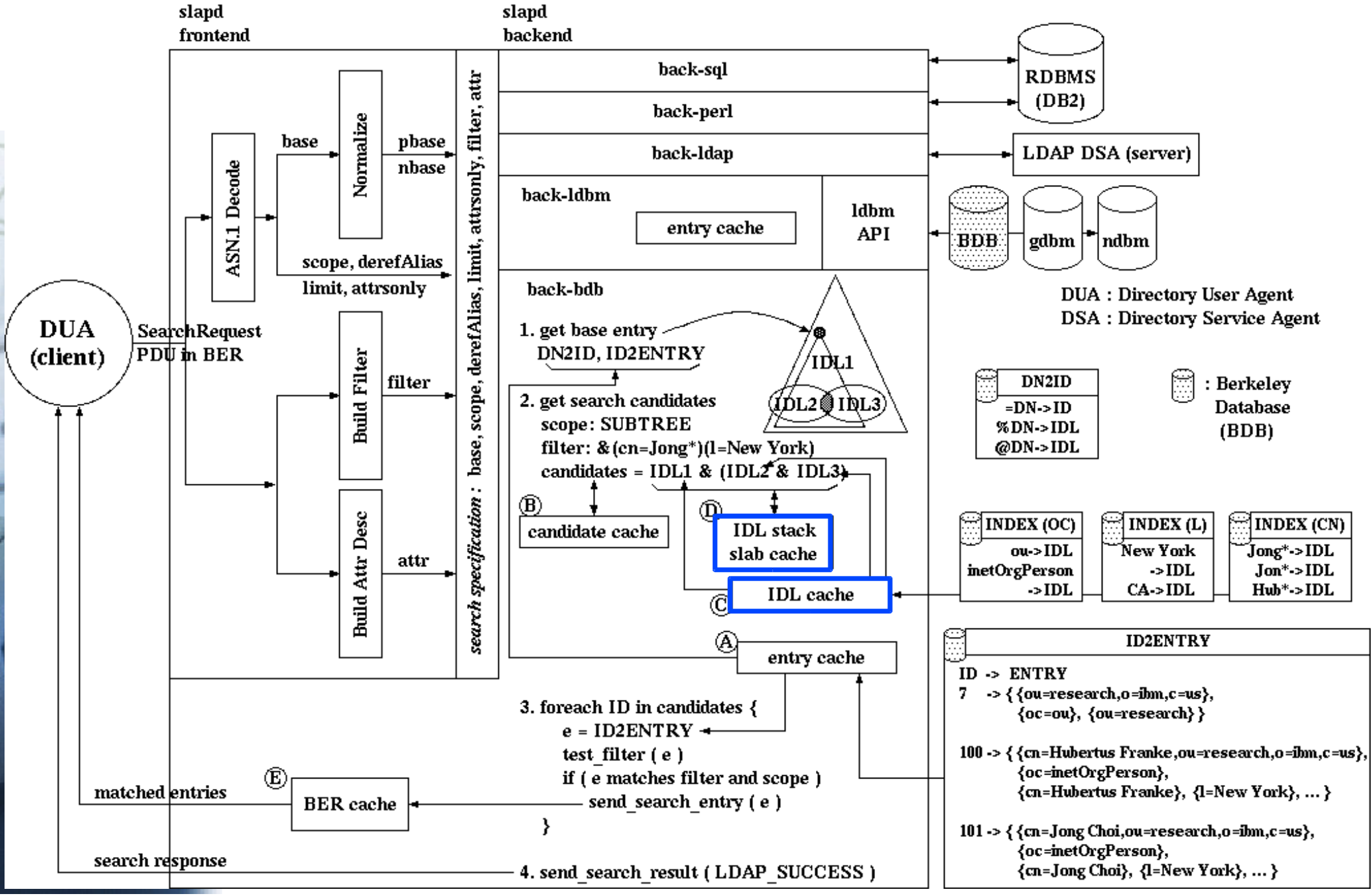
- Cache IDL of candidates of a search
  - Performs better than back-ldbm
  - DirMark (mesg) : 2378 ops/sec (13% higher than back-ldbm)

| **slapd** | 44.14% | **libdb-4.0** | 3.45% | **vmlinux** | 12.69% |
|---|---|---|---|---|---|
| send_search_entry | 2.70% | _lock_get_internal | 0.49% | save_i387 | 0.87% |
| ber_printf | 2.69% | _ham_func5 | 0.36% | schedule | 0.69% |
| avl_find | 2.27% | _ham_lookup | 0.36% | send_sig_info | 0.64% |
| ber_write | 2.24% | _ham_item | 0.23% | restore_i387 | 0.61% |
| is_ad_subtype | 2.18% | _lock_put_internal | 0.21% | do_signal | 0.54% |
| ber_put_seqorset | 1.93% | _lock_getobj | 0.14% | tcp_sendmsg | 0.51% |
| **libc** | 21.54% | **libpthread** | 13.66% | **misc** | 4.52% |

- **Consistency Problem**
  - Too costly to manage mappings of component IDLs in dn2id / index DB to candidates



# IDL Stack Slab and IDL Cache



# IDL Stack Slab and IDL Cache

## Caching Component IDLs

- Dn2id / index databases, efficient consistency maintenance

| | | | | | |
|---|---|---|---|---|---|
| **slapd** | 40.37% | **libdb-4.0** | 7.14% | **vmlinux** | 16.64% |
| ber_printf | 2.28% | _lock_get_internal | 0.69% | do_zap_page_range | 2.82% |
| send_search_entry | 2.24% | _ham_lookup | 0.67% | save_i387 | 0.76% |
| is_ad_subtype | 1.96% | _db_tas_mutex_lock | 0.46% | schedule | 0.68% |
| ber_write | 1.92% | _ham_item | 0.45% | send_sig_info | 0.61% |
| avl_find | 1.86% | _ham_get_cpage | 0.39% | get_unmapped_area | 0.50% |
| ber_put_seqorset | 1.63% | _ham_item_next | 0.35% | restore_i387 | 0.48% |
| **libc** | 19.33% | **libpthread** | 12.78% | **misc** | 3.74% |

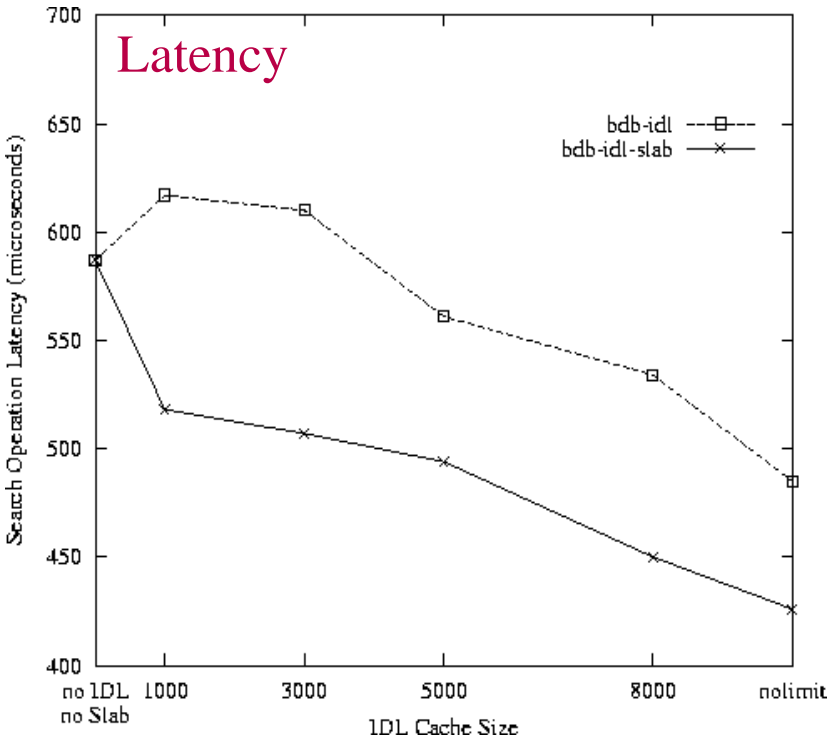
## Caching IDL Stack Slabs

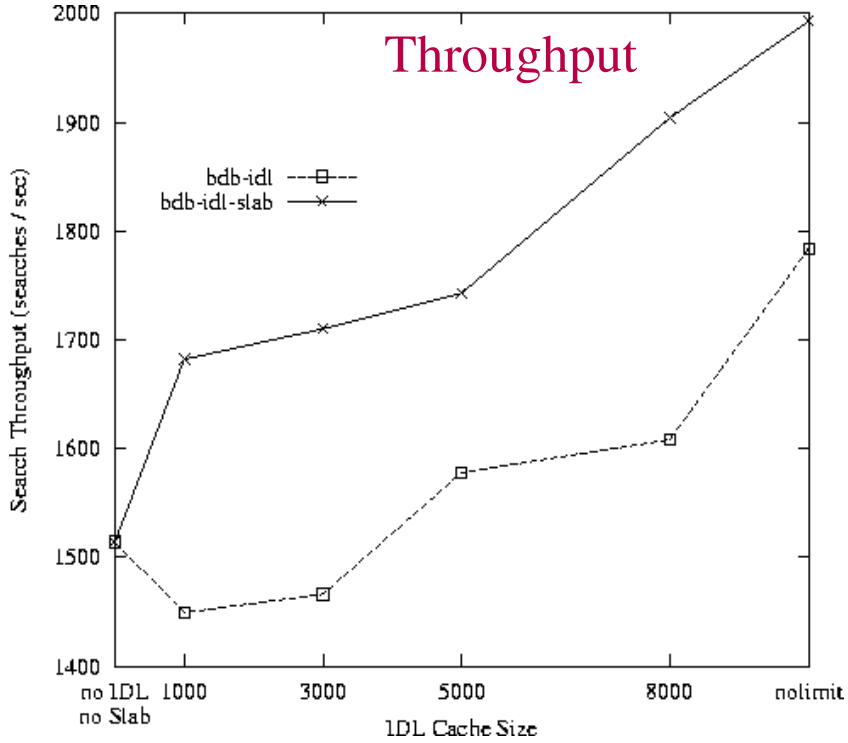
- search_candidates() allocs (depth+1)*IDL_UM_SIZE stack
- Slab cache for IDL stack, depth is fully configurable

| | | | | | |
|---|---|---|---|---|---|
| **slapd** | 43.31% | **libdb-4.0** | 7.41% | **vmlinux** | 11.43% |
| send_search_entry | 2.51% | _lock_get_internal | 0.75% | save_i387 | 0.85% |
| ber_printf | 2.39% | _ham_lookup | 0.66% | schedule | 0.74% |
| is_ad_subtype | 2.00% | _ham_item | 0.47% | send_sig_info | 0.59% |
| avl_find | 1.98% | _ham_get_cpage | 0.41% | restore_i387 | 0.52% |
| ber_write | 1.91% | _db_tas_mutex_lock | 0.38% | tcp_sendmsg | 0.48% |
| ber_put_seqorset | 1.73% | _ham_func5 | 0.38% | do_signal | 0.48% |
| **libc** | 20.43% | **libpthread** | 13.79% | **misc** | 3.63% |

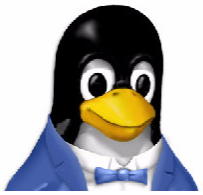
# IDL Stack Slab and IDL Cache

## Performance of Combined IDL Stack Slab and IDL Caching

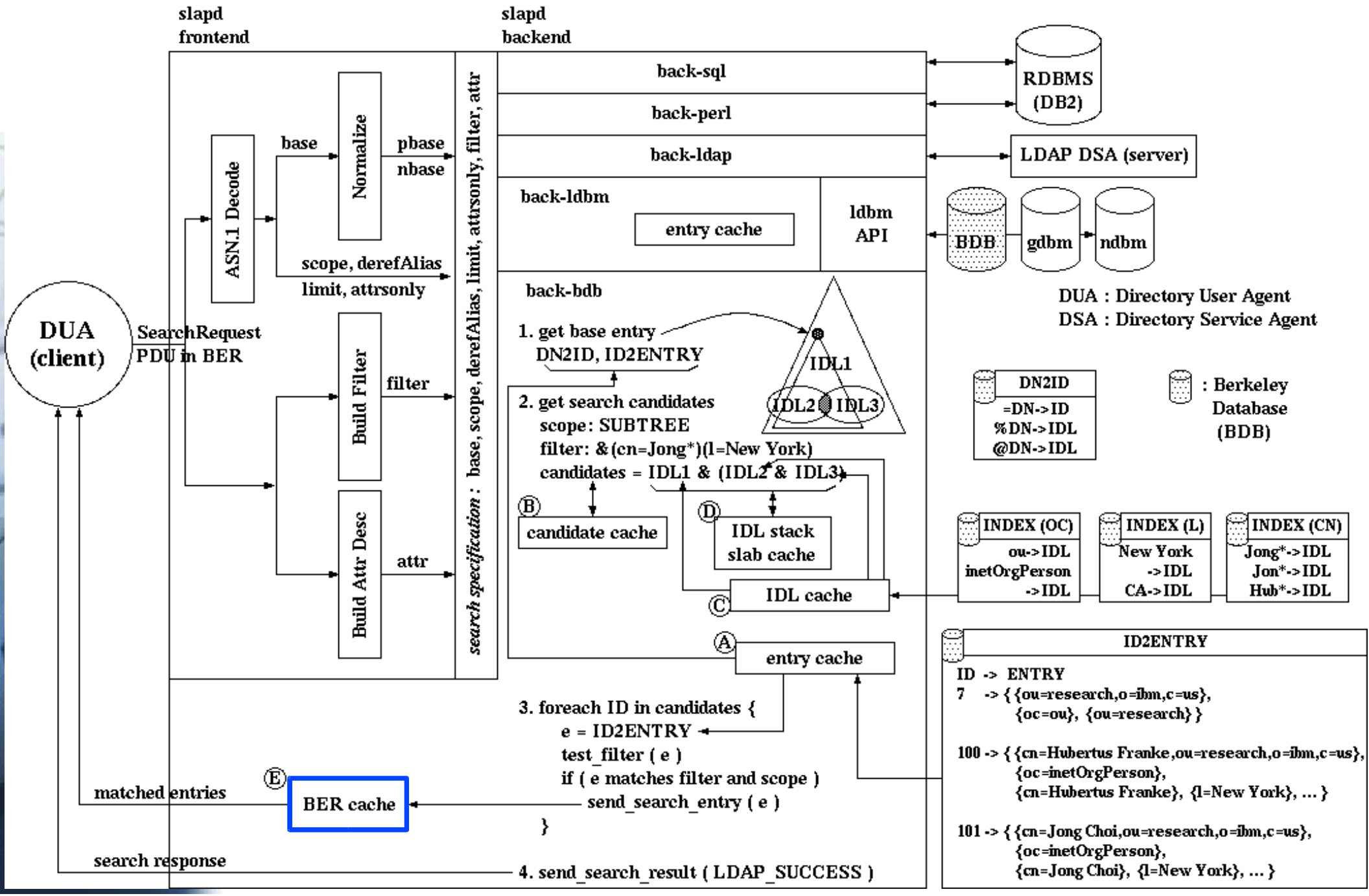



- Currenty in HEAD
- Fully configurable in slapd.conf
  - idlcachesize <cache size in # of IDLs>
  - searchstack <slab size in filter depth>



# BER Transfer Cache



# Summary

- **Back-bdb Performance Improvements (mesg)**
  - Entry cache : 1006 ops/sec -> 1520 ops/sec (50.5%)
  - IDL / IDL stack cache : 1520 ops/sec -> 1993 ops/sec (31.1%)
  - BER cache : 1993 ops/sec -> 2258 ops/sec (13.3%)
- **Further Performance Studies**
  - BER cache
  - DN cache
  - ...