Advanced Database Management

Within routers used at the metro/edge, different applications are run to implement forwarding and classification. Each forwarding algorithm and usually each traffic characteristic classified constitutes an application. These various applications utilize unique databases, which are based on the fields being used for query. Depending on the type of protocol utilized, forwarding can be implemented in a number of ways, including the use of MAC addresses (Layer 2), IP addresses employing longest prefix (Layer 3), or through recognition of some other unique set of field values in the header (flow cache). Over time, the databases will accrue unique lists of hits that apply to that application.

To streamline both the classification and forwarding tasks for metro/edge routers, designers must handle multiple databases. Dedicating a separate database to each individual classification and forwarding application enables the designer to have a much more scalable and manageable design that results in higher performance and flexibility (figure 1). Using previous generations of content addressable memories (CAMs), the designer differentiates each entry in the CAM array as a member of a different application. While this is not much of a problem with two databases, this becomes very complex as the number of databases grows and multiple widths are supported.

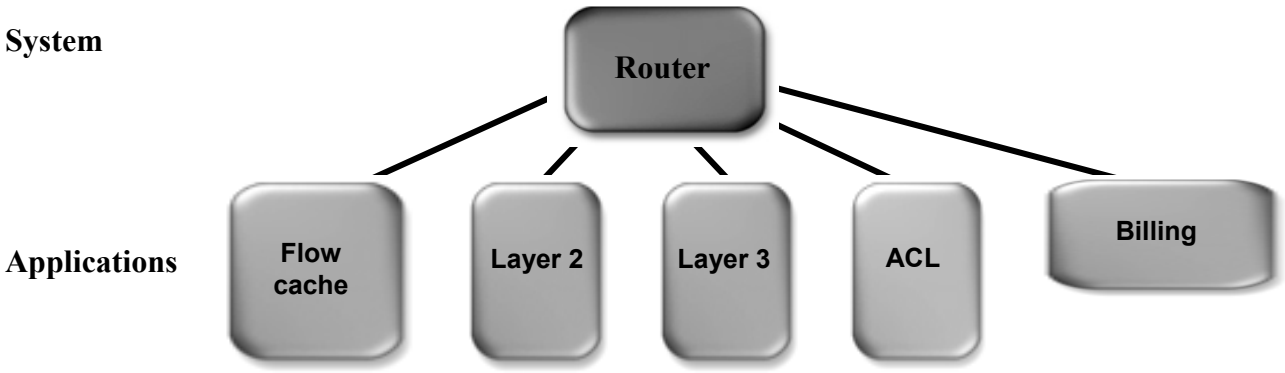


Figure 1. Multiple applications for forwarding and classification are better supported with a unique database assigned to each.

By putting the database select specification into the instruction of the co-processor rather than the database entry, search key generation becomes easier and faster (by two times in some cases). Organizing data into databases also make updates easier, since updates are specific to databases. Moreover, using multiple databases helps conserve power because only a limited portion of the overall memory is energized.

Multiple Databases Strain Database Management

As figure 2 shows, different application databases require significantly different resources for the device storing the databases, whether it is a CAM, classification engine or IP coprocessor. These requirements include the depth of array allocated to the application, size of data (that is, the width of the key), type of search desired (exact match or ternary), frequency of access, insertions and

deletions. The better the device storing the databases can match the unique requirements of all the applications using it for acceleration, the better the performance/cost ratio will be for the overall product.

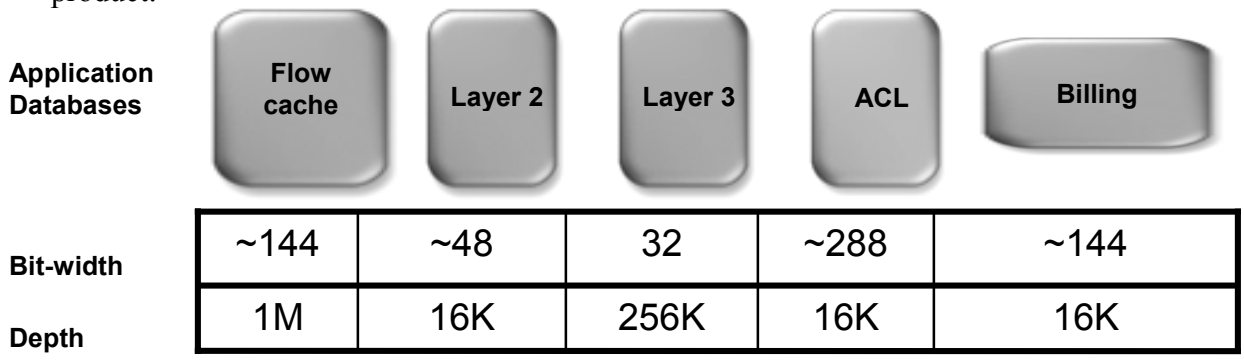


Figure 2. Database resources required for various services.

Techniques to access multiple databases for search, classification and management tasks are evolving to keep pace with the proliferation in databases. At first, CAMs had no intrinsic support for databases, so the strategy was to rely on embedded database identifiers (figure 3). However, identifiers are cumbersome to deal with, taking up valuable memory space and complicating updates and database management. For systems that have different width tables stored in one device, the database identifier bits must be replicated throughout each record to ensure that portions of longer records will not be selected mistakenly while searching on shorter keys. Besides adding overhead in generating search keys, using embedded identifiers requires that the entire device be energized when any table is searched, leading to higher power consumption.

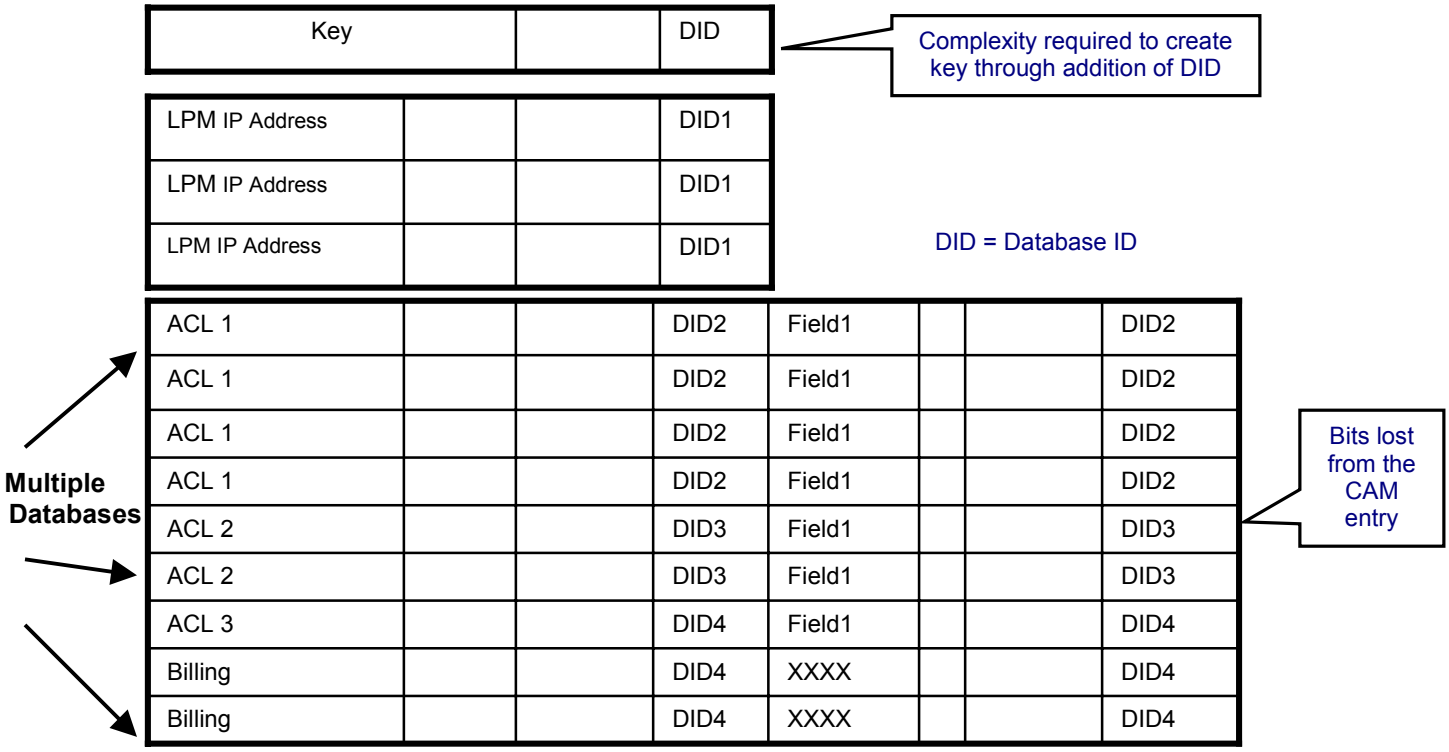


Figure 3: Diagram of database with database identifier bits.

Another method allows multiple databases to be selected via control fields contained in configuration registers. These registers can be written to, eliminating at least one instruction when the system switches between databases. With this approach, a register that selects the necessary database for an activity is placed inside the memory device. While this method alleviates the wasted data bits and power problems associated with embedded bits, the random nature of communications traffic can cause a 50 percent reduction in search performance as writes occur to the select register to access the appropriate database. Every time the system switches to a new database, it must rewrite the register, burning valuable cycles. Although it is an improvement over using embedded database identifiers, it works best with only a few databases.

Some register-based devices allow the system to select the database based on the width of the entries to support IPv4 and IPv6, enabling the CAM device to support multiple widths simultaneously. With this technique, only those entries that correspond to the width as selected by the search command will participate in the search. One drawback with these devices is that not all applications can be differentiated by width. Classification applications often require multiple databases of 144 bits or like wide entries. Again, embedded bits will be required in these cases to differentiate between these databases. This method does reduce the number of wasted bits to duplicate database identifier bits on wider entries; however, the duplicate field trade-off is not alleviated. Power consumption and application software are not optimized because more data than is required participates in the searches and the host must control this challenge.

In addition to the aforementioned benefits of performance and efficiency, employing the database approach also reduces the need for global mask registers. Besides handling embedded bits, the programmer often has to create additional masks to achieve the functionality of native database support.

IP Co-Processor Dynamic Database Management

These problems are driving the need for a next-generation approach that takes into account the emerging reality of managing multiple databases of varying sizes. IDT's family of IP co-processors supports multiple databases in *hardware* as opposed to software. The instruction format includes a field for selecting a database, thus a new database can be dynamically selected on each and every operation of the co-processor. This field is directly embedded within the instruction itself and is large enough to support upwards of 16 databases. Now it is possible to easily create and access multiple databases, each of which can be any width the designer specifies, thus enabling dynamic database management.

Dynamic database management also dramatically streamlines multiple database searches, since it allows for single-cycle selection between multiple databases. There are structures within this approach that help narrow the search to the actual entry, energizing only the appropriate area in the database to further conserve power consumption and speed access to the data. This refined selection function removes potential overhead when creating the selection key. It also minimizes the number of entries required to support selection while reducing the mask register requirements. The result is less power consumption, faster database selection and entry insertion, and increased functionality with better utilization of database entries.

Minimized Power Consumption
Such a flexible, efficient approach for accessing multiple databases yields additional benefits in power consumption. The power consumption is a function of the specific IP co-processor operation—the word width, the clock frequency, the duty cycle and the number of entries enabled. The worst-case situation occurs when performing a x72 or x144 lookup with 100 percent duty cycles on all entries. To help designers achieve the desired power consumption for a specific application, IDT provides flexible power management features in the IP co-processor. One of the most significant features is the device's ability to power up just the selected database needed for a particular IP operation. This capability significantly minimizes power consumption without affecting the performance of the IP co-processor. In applications where large forwarding databases exist in conjunction with content classification databases, this savings can be in the order of 66% of the worst-case power figure.

Streamlined Database Maintenance
This next-generation dynamic database management also helps streamline important database maintenance activities. Database maintenance involves a variety of responsibilities including:

- Creating/deleting a database
- Adding new entries
- Deleting old entries
- Modifying an entry, including modification of an entry's associated data (if there is any)

These tasks occur in response to a system change or update. For example, there might be a BGP routing table update or perhaps the detection and/or classification of a new packet flow. Or the user might add, delete or update filtering criteria via the management interface. In short, database management not only includes management of an entry's associated data, but where and when an entry is manipulated based on a specific application or implementation.

Maintenance requirements depend on and are limited by the plane in which they are executed. For more information on the network planar architecture used in most routers, go to the Database Architecture for Packet Processing sidebar. Typically, database maintenance responsibilities are assigned to only one plane based on the content and management requirements of the database. However, database access may occur from both planes simultaneously. The control plane usually assumes responsibility for maintenance because it can do significantly more complex operations without impacting the packet processing. The data plane needs to focus on processing packets and works within very tight timing constraints to achieve real-time processing. However, there are times when it makes sense to use the data plane. NPU data plane database management must be tightly written, as instruction space is limited.

Exact Match Database Basics
In an exact match database, all the bits of an entry must match, such as the source and destination IP address and the source and destination UDP/TCP port number. All entries have equal priority and only a single match result possible in the database because the entries are very unique.
Exact match databases are used for Layer 2 MAC tables, static address access lists (ACL), dynamic ACL for state-based security, or flow caching.

The type of maintenance performed depends on the database. There are three basic categories for databases: exact match, longest prefix match (LPM), and classification databases. Each requires different capabilities to meet its unique maintenance needs. The IP co-processor from IDT provides

software libraries for maintenance through the control plane, supporting a broad range of functionality including:

- Non-blocking entry addition
- Entry deletion by both key and entry address
- Entry modification
- Associated data management

Exact Match Database Maintenance: Exact match databases are used for Layer 2 and flow cache forwarding. Flow caches are highly dynamic with update rates of 100 Kbps. The quickest, most efficient way to update the database is from the data plane. It is important to minimize NPU utilization so as not to impede the overall packet processing. The ideal case is only a single instruction for updating the exact match database.

Learning and aging capabilities are important for keeping entries as fresh as possible. Learning is the process of creating and updating entries to the database. Aging occurs when an entry in the database has not been accessed within a defined period of time. During learning, new entries are added to any available empty location in the database. The database library creates a new entry by finding any empty location and storing the new data in that spot. Another approach is to rely on a next-free address (NFA) register that contains the address of the next empty location. Any writes or deletions to the database will require an update of the NFA register. The IP co-processor contains multiple NFA registers that can be associated with individual databases, compared to just one NFA register in earlier generations of CAMs.

Often designers want unused data to be eliminated, freeing up space for new entries. Aging is a mechanism that allows the exact match database to stay current with the volatile nature of the network. It is used to remove stale entries and free up more space as the network topology changes over time. Since the exact timing of aging is a non-critical, low-priority task, it is done in background when the data plane has a free moment to perform this function. The aging task must minimize NPU intervention and overhead to ensure high-performance packet processing.

Longest Prefix Match (LPM) Database Maintenance: Update rates for an LPM database are about 5 Kbps. Updates come from advertisements sent between routers. For that reason, the control plane typically maintains this type of database. The most common form of LPM usage is for classless inter-domain routing (CIDR), developed to address the issue of different size Internet addresses. CIDR can accommodate addresses of any length, relying on masking to identify the important bits in an address. The longer the mask, the more precise the address and consequently the better the choice of the route. Full ternary capabilities (1, 0 and don't care) are employed in implementing CIDR.

LPM Database Basics

Entries in a LPM database are grouped based on the number of consecutive ones in their subnet mask (i.e., a weight or weight class). The same weighted entries all have equal priority, so the order within the weight does not matter. However, the position of a prefix weight *class* is important. The highest priority corresponds to the largest weight value.

The longest prefix length is given the highest positional priority. For example, an IPv4 address would have the highest priority with a /32 prefix because all the possible bits in the address are used.

Examples for LPM database include classless inter -domain routing used for IPv4 forwarding and routing.

The biggest challenge with a LPM database occurs when the database starts getting full. This is becoming a huge issue as routing tables grow by leaps and bounds (figure 4).

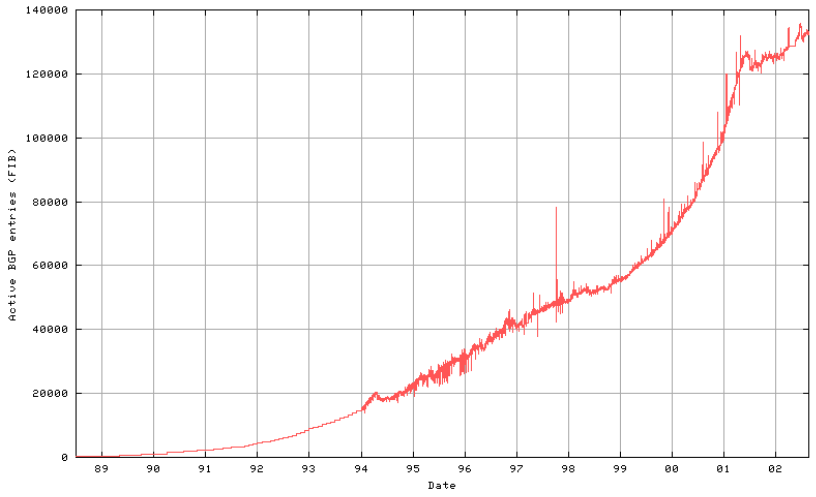


Figure 4. Growth in BGP entries is driving up the size of databases.

Not only are there more addresses, the prefix distribution also is not equitable (as shown in figure 5). Note that the scale in this figure is logarithmic. There are only a few entries for each of the prefixes on the ends, but /24 has the majority of entries. As a result, the prefix distribution in the database is non-linear and the prefix groups must be allocated relative to the size of the number of entries in the prefix grouping.

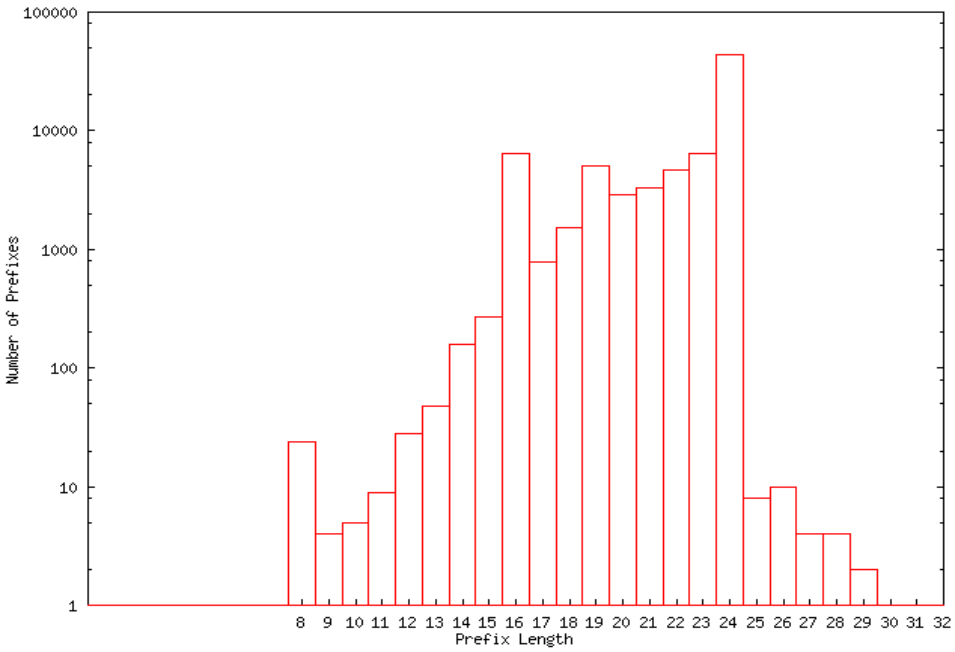


Figure 5. Prefix distribution (April 11, 2000).

With the incredible growth in IP addresses, designers must create solutions with room to grow, hence the need for forwarding tables that can hold 512K entries even though the table size today may be 120K. Because the network is dynamic due to constant expansion and fluctuations in connectivity between subnets, the forwarding table must be able to respond to rapid route updates. For this reason a key requirement of a total system must be how table maintenance is done.

When initializing a forwarding information database (FIB) that addresses CIDR, the CAM is allocated into prefix weight classes. For a database that represents the Internet, the database must be able to allocate 24 weight classes with different sizes. As figure 5 indicates, approximately 30K entries are needed for /24 prefixes. Another eight of the weight classes—from /16 to /23—require about 10K entries each. The remaining 15 weight classes will only need around 1K entries.

Requirements as a Result of BGP Protocol

Additions and deletions to the FIB are made as a result of routing protocol updates through the control plane. Routing protocols like BGP are used to announce and withdraw entries in the routing information database (RIB) as a result of changes in the network. The BGP protocol always deletes entries before adding new entries. Furthermore, new announcements of additions are limited to one-per-BGP update packet, whereas multiple withdraws can arrive in one BGP packet. This is a reflection of the fact that deletions should happen fast, while additions need not happen as quickly. In effect, the network works best when packets destined for non-existent networks are terminated quickly. The order and inherent ratio of withdraw-to-announcements indicates that most of the time, there will be room to add new entries into the CAM.

Routing update measurements collected by monitoring MAE-West and MAE-East show that even in the Internet, the largest network in the world, updates do arrive more often than 100 per second. One can well imagine that in an enterprise network or a managed, private wide area network (WAN), the number of updates will be less. Only in the worst conditions, occurring once or twice per month on the Internet, is there a condition known as a route flap, which creates as much as 3K announcements in a second.

Addressing Worst Case Conditions

When a prefix becomes full, another prefix with a free space must be found, and that space must be re-allocated to the full prefix. Therefore, the challenge becomes how to move the entry into the desired weight class quickly and efficiently. Some vendors offer costly hardware-based solutions for LPM database maintenance adding to silicon cost and complexity. An approach offered by IDT's IP co-processor family is to implement portions of the LPM maintenance drivers in software that meets and exceeds update requirements of protocols like BGP. Not only is this method cost-effective, more importantly, it offers greater flexibility in database maintenance as more and more services and addresses are added.

When adding a new entry that is full and has no empty slots to a LPM database prefix, the software driver in the IP co-processor follows these steps:

1) Locate the nearest weight class with a free entry
2) Move the free entry to the lowest address of the source weight class where it currently resides

3) Transfer the free entry to the adjacent weight class by moving the contents of the lowest address of the adjacent weight class to the lowest address of the previous weight class (where the free entry was moved)
4) Continue transfer or "shuffle" the free location in this manner until it reaches desired weight class where it is needed

[This example assumes the free entry is at a higher address than where it is required. If the free entry was at a lower address then the moves would be to highest addresses in the example.]

The worst-case number of moves is one half the number of prefixes (12 in the case of IPv4 CIDR). On the average it will be half the worst case resulting in six moves. The worst –worst-case scenario comes when a piece of equipment is nearing its useful life and is about to be retired from service. Just like a PC whose disk drive is nearing full, the performance will drop. This is because the free space may be at one end of the table that is against the limit of the CAM, thus now free space. In this rare case, when the equipment is about to be removed from service, the worst-worst-case number of moves will be equal to the number of prefix weight classes, thus only doubling the requirement for the most conservative design. However, as this scenario is only relevant when the equipment needs to be removed from service and represents a gradual degradation, we will not consider it.

One of the concerns about implementing database maintenance in software is the impact on overall system performance. Performing the IP co-processor shuffle in a shadow DRAM dramatically optimizes entry creation in a LPM database. IP co-processors, utilizing CAMs, are optimized for searches and writes, and not for a reads. Consequently, shadowing the LPM database in DRAM and performing the shuffle in that environment provides the best performance. The DRAM image is then treated like a write-through cache and only the results of writes are transferred back into the IP co-processor, eliminating more than 90 percent of the overhead impact on the data plane for the LPM maintenance task. In fact, the IP co-processor shuffle results in a worst-case impact of just six percent of the overall system CPU budget.

To appreciate the low impact, the IP co-processor shuffle has on system performance, it helps to look at the LPM database maintenance best-case scenario versus the worst-case scenario versus the worst worst-case scenario. For the best case, assume there is a free space in the specific weight class where the data needs to be stored, resulting in one write per insertion. Based on a rate of a 100 inserts per second, the overall rate for LPM maintenance is 100 writes per second. In the typical worst-case scenario, on the other hand, assume there are no empty slots in the desired weight class. One insertion will cost require a free entry being shuffled through half of the weight classes—in other words, twelve shuffles. For example, if a route flap causes 3K inserts in one second then the typical worst case is 36K writes per second.

Using these numbers and assuming a shadow-based control plane approach with an IP co-processor, it takes approximately 500 additional instructions to move a free entry. When assuming 100 MSPS, only 0.009 percent of the IP co-processor bandwidth is consumed to support each additional move of a free entry.

Applying this to the typical and worst-case yields the CPU requirements needed to support LPM database maintenance with the IP co-processor shuffle. In the average case, there are:

100 writes x 500 instructions x 6= 300 KIPS

The worst-case scenario with a 3K-route flap requires approximately:

3K x 500 instructions x12 = 18 MIPS

The latest generation of NPUs will have 64-bit CPUs for maintenance that will deliver 600 MIPS. Even with a 300 MIPS host process the worst-case load will be much less than one percent of the overall system CPU budget. As host CPUs get faster, this number will only decrease.

Classification Database Maintenance: Maintenance for the third type of database is quite different than for LPM and exact match databases. With the classification database, all entries are hand entered, typically by the IP manager, and are updated quite infrequently. In fact, it is considered a near-static database with very low dynamic update rates on the order of one change per second. Consequently, database maintenance is not a pressing issue with this type of database structure.

Rules in the classification database are usually broken down into at most eight entries. Since each classifier only has a few rules, the likelihood of needing to move a free entry is not very often. But in the rare times when it needs to happen, the nearest entry can be "shuffled" into position using the same approach as discussed in the LPM database maintenance section. Figure 6 summarizes the impact of maintenance on the various types of databases.

Classification Database Basics

A classifier or policy database is composed of a set of rules or policies. Entries are ordered based on a set of criteria. Each rule or policy can be decomposed in a set of entries. Order is important to the rules and the entry decomposition and must be maintained for both.

Decomposition can be a CPU-intensive task and is typically offloaded to the control plane. This type of database is heavily ternary.

Examples include static access control lists, like billing or policy-based routing such as traffic flow control and QoS. Other types of classification databases include dynamic ACL, including dynamic QoS and exact match databases.

	Forwarding			**Classification**
	L2	L3, LPM	Flow Caching	Static ACL
Dataplane Ops/sec	100/sec Learn&age	not required	100K/sec Learn&age	not required
Control Plane Ops/sec	init only	Worst case 3K adds/sec (writes from BGP)	init only	1/sec (written from list)
Host CPU Maintenance (600 Mhz 64-bit CPU)	not required	Typ. < .1%	not required	~ < 0.1 %
		worst-worst 3%		
NPU Maintenance	100 * (Lookup&Store)	not required	100K * (Lookup&Store)	not required
CAM B/W Maintenance	~0%	0.010%	1%	~0%
CAM Entries	16K	512K (BGP)	1M	16K
Best Handled through:	CAM H/W	S/W driver	CAM H/W	S/W driver

Figure 6. Maintenance performance summary.

Summary

High performance communications systems are not just handling high bandwidths of data, they are running multiple applications that determine what the content is, what special services it requires, how to account it, and where to send it. These applications each have unique lists of rules, or user data that are required to be searched efficiently to handle the traffic and quickly forward it. These are recognized as separate databases of data.

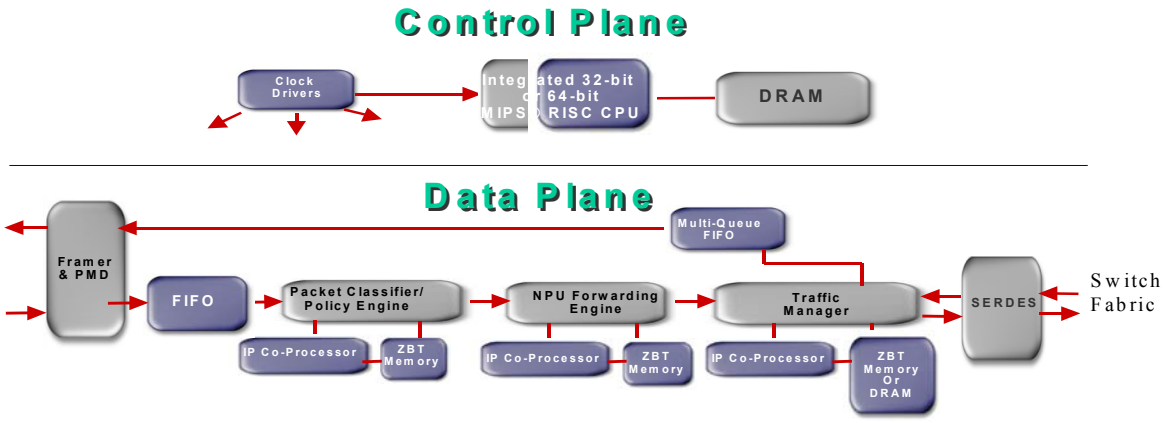
CAM-based IP co-processor devices provide an ideal mechanism to accelerate the recognition of the various traits of the traffic. In packet-based networks, this allows more value-added activities to be supported in the latency before the next packet is expected.

The databases associated with different applications are stored in the IP co-processor for quick searching. Factors important to the utilization of CAM technology are density, power, and performance. Performance at the system level with real traffic requirements is an important consideration. The handling of the databases within the CAM sub-system directly affects each of these criteria. The ability to select which database a CAM operation is to apply allows efficient use of the memory resources available to the user and provides more of the advertised density for the search activity. Further, power is significantly reduced when only data associated with particular database activities participate. In applications where large forwarding databases exist in conjunction with content classification databases, this savings can be in the order of 66%. An effective selection mechanism allows these benefits while at the same time simplifying the host formatting of the search key. This results in the improved performance at the CAM also being passed along to the host processor as simplified application software, and improved processing bandwidth.

When designing a packet processing solution it is important to pay attention to what tasks are critical, and/or dominate in terms of frequency, thus dominating the performance characteristics of the system. IDT's dynamic database management optimizes database searching, one of the key tasks in packet processing. While it is important to be able update a database in a timely fashion, the designer must take into consideration how often that task is to be performed. Keeping this in mind, the IDT IP co-processor family enhances the system solution by providing a tuned library of code to perform database maintenance. The combination of hardware dynamic database management, and optimized maintenance support libraries results in a solution that is optimized for performance, cost, and time to market.

Sidebar: Database Architecture for Packet Processing

Most routers and switches incorporate the classical network planar architecture model to govern interactions with databases and the outside world. This architecture is comprised of three planes—the management, control and data planes.



The management plane supports the system administrative interface including configuration, provisioning, and monitoring via methods including simple network management protocol (SNMP), CLI, Web servers, etc. Generally written in C/C++, it includes the graphical user interface (GUI) presented by the equipment and runs on a general-purpose processor. The management plane is the low-priority, low-bandwidth, low-change path as compared to the other planes of the system.

The control plane contains the general-purpose software responsible for managing the whole packet-processing operation, including the data planes behavior. This plane is responsible for processing the data plane "exception" packets/conditions. A data plane exception packet may be any of the following: routing update packets from BGP, RIP, OSP; signaling protocol packets such as MPLS, RSVP, LDP, Q2931; discovery protocol packets such as ARP, ICMP, IGMP; and informative protocol packets such as ICMP. It may also include new packets for flow classification associated with the first-time packet observation.

The engine in a control plane is typically a RISC processor, which may or may not be contained within the NPU device responsible for the packet processing. The control plane contains large CPU instructions sets and is usually written in "C" code, and therefore has a larger instruction space and a more generalized instruction set. This plane can take on much more complex database management operations. There is more time to execute management tasks since the control plane is nondeterministic; it can accommodate out-of-band additions, deletions and modifications to the database and has loose (but finite) bounds on database processing time. Consequently, performing

complex management tasks from the control plane does not affect the packet-processing throughput.

The data plane represents the hardware and high-performance software {microcode/picocode}, performing line-rate per-packet operations. This is the critical high-speed, high-performance path for performing real-time, in-line packet processing. The databases reside on the data plane, containing the information needed for the various services the router supports.

The data plane is a highly optimized entity designed to maximize throughput, with the data plane engine specifically tailored to the forwarding task. Typically, these are NPU microengines located in the NPU component. The microengines are the entities that contain the specific threads/tasks/contexts that operate on each packet. All code is written in native assembly language, customized to the implementation and optimized for specific packet-processing tasks, including packet modification.

The data plane is governed by deterministic and bounded maintenance time requirements. Each packet has a finite processing time and searches must be completed in a bounded time in order to be optimized in this plane. Typically, a packet is only allowed a few instructions (20 to 50) before needing to be passed on. Thus, every instruction execution is critical.

Planar Support for Database Management

Maintenance operations (add, delete, modify) are almost always executed from the control plane and in a few cases from the data plane. Typically, database management responsibility is assigned to only one plane, based on the content and management requirements of the database. However, database access may occur from both planes simultaneously.

The control plane can take on much more complex database management operations because it is usually written in "C," and therefore has a larger instruction space and more generalized instruction set. Additionally, there is more time to execute management tasks since the control plane is nondeterministic. Consequently, performing complex management tasks from the control plane does not affect the packet-processing throughput.

There are times when it makes sense to use the data plane. For example, use of the data plane is appropriate to add entries and specific CAM instructions developed to simplify and speed up these operations. NPU data plane database management must be tightly written, as instruction space is limited.

As more and more services are added and as data rates move towards OC-768, classification and forwarding are becoming much more complex tasks. As result, more databases are needed to support these services, placing a greater burden on database management and maintenance. IDT's IP co-processors provide next-generation capabilities that are designed to support the high-performance, policy-based requirements for future applications that will undoubtedly have numerous databases running at OC-192 and OC-768 line rates.