Quiz on Many-core Systems

Test your understanding of concepts related to the design of a many-core key-value store via a quiz.

Question 3

Our designed key-value store helped in storing much more keys in one node. Is that not a bad idea when a node fails since it will take down a large portion of the keys?

Hide Answer

It is a trade-off between key density in a node versus fault tolerance. If we pack more keys in a node and it fails, it reduces the availability of all those keys. If we keep a few keys in a node, we will not be utilizing expensive resources efficiently.

Usually, key-value stores are used as in-memory caches for data. On failure, we will need to refill this cache on a new node.

We encourage you to think about how you will balance this trade-off between key density and fault tolerance.

Question 1

In the context of the key-value system, why was the domain-specific system better than the general-purpose x86 node?

Hide Answer

There are multiple reasons why we've preferred a domain-specific system over the general-purpose system:

- Good performance per watt
- Better opportunity for parallelization due to the availability of many cores

Question 2

What are some of the cons of using a domain-specific system?

Hide Answer

Oftentimes, a domain-specific system has many peculiarities that are not found in a general-purpose system. Some of these can be as follows:

- Small virtual address space and larger physical addressability.
- Needing to change the software to benefit from the system fully.
- Absolute processor performance is lower than a typical general-purpose system, hindering some use cases.

Question 3

Our designed key-value store helped in storing much more keys in one node. Is that not a bad idea when a node fails since it will take down a large portion of the keys?

Hide Answer

It is a trade-off between key density in a node versus fault tolerance. If we pack more keys in a node and it fails, it reduces the availability of all those keys. If we keep a few keys in a node, we will not be utilizing expensive resources efficiently.

Usually, key-value stores are used as in-memory caches for data. On failure, we will need to refill this cache on a new node.

We encourage you to think about how you will balance this trade-off between key density and fault tolerance.