their database model and in consequence send a new network data to SESC on average every week or so. Using the conventional ONLINE-STANDBY architecture for the SESC is not feasible. A single node based on this configuration is not very scalable and will result in a single point of failure that we would like to avoid. In our work we propose a different approach for the design of a SESC.

V. CLUSTER COMPUTING

In our approach instead of using ONLINE-STANDBY architecture we propose to use a distributed environment of many computers connected together. This type of architecture will provide very good scalability. Should we need to provide service to more utilities we just need to add more computers to our network. In addition our architecture will not suffer from single point of failure. We propose that all SCADA data coming from the field and all data related to SE solution is stored in the memory of the distributed computers. This is essential to provide fast response to requests issued by the power utilities and is the core of our design.

The proposed architecture is known as Highly Available Distributed RAM (HADRAM). In our design we harness the capacity and power of distributed main memory of individual computers that is highly available and easily recovers from node unavailability. The high level architecture of the SESC is shown on Figure 2.

HADRAM is made up of a cluster of computers all connected together at the SECS. The SESC may or may not be at a single physical site. Every computer node in the system can be either a server or parity server or both. All the input data sent to the SESC and the calculated results including the SE solution, is stored in the memory of the servers. Every utility has therefore its own dedicated in-memory database at the SECS.

In HADRAM architecture, memory is reserved for a given utility's database in units of memory blocks called H-blocks. Every computer node has several H-blocks that might belong to one or several utilities.

Some utilities may have their H-blocks distributed over several servers. This allocation may change dynamically over time to allow for load balancing and/or during recovery from a failure of a server. The shaded boxes in Figure 2, represent already reserved H-blocks. Every server has the capability to run SE function and calculate the SE solution. During normal operation a power utility is connected with a SECS designated server. Its data is stored on several nodes and one node is designated to be the main node for the given utility. HADRAM can provide an arbitrary level of failure tolerance against component failure, while still retaining the very attractive performance of distributed memory (access times at the order of network latency). In addition, HADRAM system provides the low cost of using commodity rack servers instead in-memory database as a service, is how to avoid loss of data if one of the nodes that contain data for a given utility becomes unavailable?

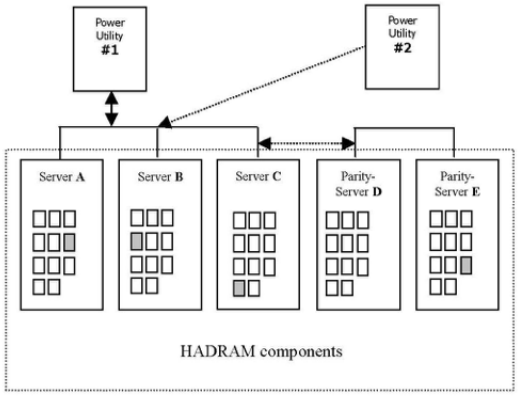

Figure 2: SESC architecture

A. HADRAM Reliability Mechanism

Distributed storage needs to protect its data against component unavailability [2],[15]. The only possible method is to store data redundantly. There are several ways to accomplish this task. The simplest solution, replication of data (used in current EMS architectures), is however very costly. To achieve k-failure tolerance, we need to replicate each piece of data in k+1 components, thus using k+1 times as much storage as if we did not provide failure tolerance. A better and more efficient storage solution is exemplified by Redundant Arrays of Inexpensive Disks (RAID) Level 5, where m data blocks on m different devices are grouped together in a *reliability stripe* to which another block – the parity block – is added to contain the bitwise parity of the m data blocks. The storage overhead is now $1/m$, but the system is only 1-available. To achieve higher availability, one can either put a single data block in different reliability stripes or one can add more parity blocks to a single reliability stripe as in the RAID Level 6 organization. The parity data is then calculated with an erasure correcting m out of n code. The erasure correcting property of such a code means that all data can be reconstructed as long as we have m out of the $n = m + k$ blocks in a reliability stripe (obtained by adding k parity blocks to the m data blocks).

In our implementation we use generalized Reed Solomon (RS) codes. Parity calculation and data reconstruction involves calculation in a Galois field with 2^f elements, where $f = 8$ is a simple, canonical choice. Our implementations of these parity calculations [6],[12] in the setting of highly available