Brief Announcement: A Fast Distributed Slicing Algorithm[*]

Vincent Gramoli
EPFL - Univ. of Neuchâtel
vincent.gramoli@epfl.ch

Ymir Vigfusson
Cornell University
ymir@cs.cornell.edu

Ken Birman
Cornell University
ken@cs.cornell.edu

Anne-Marie Kermarrec
INRIA
anne-marie.kermarrec@inria.fr

Robbert van Renesse
Cornell University
rvr@cs.cornell.edu

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Network**]: Distributed Systems -*distributed applications, network operating systems*;
General Terms: Algorithms, Experimentation, Performance, Theory
Keywords: Slicing, Churn, Large-Scale

1. THE SLICING PROBLEM

Many studies have identified heterogeneous distributions of attributes (e.g., storage, bandwidth, load, uptime) in distributed systems. Applications may suffer from ignoring such heterogeneity, as already testified by benefits from super-peers/regular-peers classification. As a step further, partitioning the system into portions or slices of nodes that share similar attribute values provides a finer granularity of control.

A *distributed slicing* protocol organizes n nodes into k equally balanced *slices* in a one-dimensional attribute space, such that when the classification algorithm terminates, each node i knows its slice number j (i.e., node i knows that its attribute value is in the j^{th} portion of the largest attributed values). Potential applications include: ***(i)*** construction of multi-level hierarchies (generalizing the two-level ones common today); ***(ii)*** load-balancing in datacenters, or to provide varying levels of service in accordance with a classification of clients.

This paper introduces a simple slicing protocol called *Sliver* (for *Slicing Very Rapidly*). Sliver is accurate in that it slices static systems, unlike the Ordering protocols [1,3]. It also slices rapidly despite potential skewness of attribute value distribution, unlike the Ranking protocol [1]. The full version of this paper [2] ***(i)*** shows the benefits of Sliver on alternative techniques (e.g., leader-based approach, parallel sorting), ***(ii)*** proves convergence properties theoretically; and ***(iii)*** experimentally validates its performance in realistic settings.

The system consists of n connected nodes. Nodes can join and leave (or fail), hence the value of n changes over time. Each node i is uniquely identified and has an *attribute value* $x_i \in \mathbb{R}$ that represents its capacity in the metric of interest, for example bandwidth. At any time, the *position* r_i of node i is the index of x_i within the sorted attribute values, normalized to fall within the range $[0, 1]$. Let $P_j = (\frac{j-1}{k}, \frac{j}{k}]$ for any j, $0 < j \leq k$ denote a *slice*. Node i belongs to slice P_j if $r_i \in P_j$. Given a value of k, a slicing protocol [1] computes k slices such that each node knows its slice.

[*]This work was supported, in part, by NSF, AFRL, AFOSR, INRIA/IRISA and Intel Corporation.

A node i is *stable* at time t if it knows its slice number within at most one from time t onwards. This slack ensures that nodes whose attribute value lies on the boundary of two partitions stabilize.

2. SLIVER: SLICING VERY RAPIDLY

Sliver is a simple distributed slicing protocol that works by sampling and temporarily retaining attribute values from the network, along with node identifiers over a prespecified time window. The code run on each node is as follows.

- Each node i sends x_i to c random nodes.
- Each node i keeps track of the values it receives, along with the sender and the time they were received; discards value records that have expired and sorts the m remaining ones. Suppose B_i of them are no greater than x_i.
- Each node i estimates its slice number as the closest integer to kB_i/m.

The timeout ensures that the amount of saved data is bounded, and provides tolerance to churn. Conceptually, Sliver is similar to the Ranking protocol [1]. They differ essentially in that nodes in Sliver track the node identifiers of the values they receive, whereas nodes in the Ranking protocol only track the values themselves. This change has a significant impact: Sliver retains the simplicity of the Ranking protocol, but no longer requires uniformity in the choice of the communicating nodes.

Theorem 1 *In a static system, we expect all* n *nodes to become stable with high probability after* $O\left(\sqrt{\max\{k, \log n\} \log n}\right)$ *time.*

It follows that for $k = O(\log n)$ this bound is $O(\log n)$, and for $k = \Omega(\log n)$ it is $O\left(\sqrt{k \log n}\right)$.

To validate our result, we compared Sliver and the Ranking protocol on a 60 machine testbed using a real storage trace provided by Microsoft. Sliver converges rapidly for various numbers of slices, whereas Ranking sometimes fails to converge. A second experiment uses a trace from Skype, and confirms that Sliver can track slice numbers even in large and highly dynamic settings where nodes churn and attributes evolve rapidly.

3. REFERENCES

[1] A. Fernández, V. Gramoli, E. Jiménez, A.-M. Kermarrec, and M. Raynal. Distributed slicing in dynamic systems. In *ICDCS*. IEEE, 2007.

[2] V. Gramoli, Y. Vigfusson, K. Birman, A.-M. Kermarrec, and R. van Renesse. Sliver, a fast distributed slicing algorithm. Technical report, Cornell University, 2007. http://lpd.epfl.ch/gramoli/doc/pubs/TRSliver2.pdf.

[3] M. Jelasity and A.-M. Kermarrec. Ordered slicing of very large-scale overlay networks. In *P2P*. IEEE, 2006.