The Spread Toolkit: Architecture and Performance

Yair Amir, Claudiu Danilov, Michal Miskin-Amir, John Schultz, Jonathan Stanton

The Spread toolkit is a group communication system available from www.spread.org. Spread provides a range of reliability, ordering and stability guarantees for message delivery. Spread supports a rich fault model that includes process crashes and recoveries and network partitions and merges under the extended virtual synchrony semantics. The standard virtual synchrony semantics is also supported.

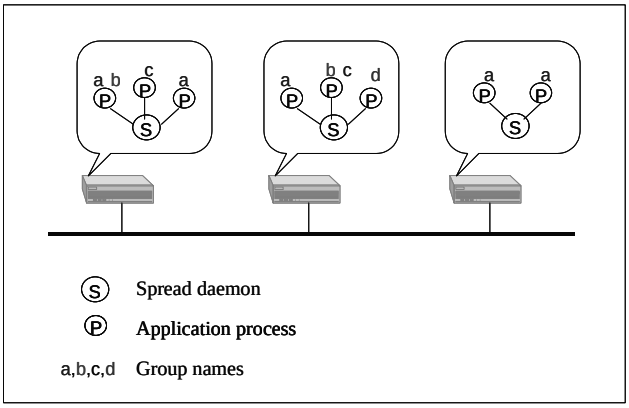

Figure 1.1: The Spread Client-Daemon Architecture

Spread is highly configurable, allowing the user to tailor it to their needs. Spread can be configured to use a single daemon in the network or to use one daemon in every computer running group communication applications. Figure 1.1 presents a case where each computer executes one Spread daemon. As can be seen in the figure, all the physical communication is handled by the daemon. The Spread daemons keep track of the computers' heavyweight membership. Each daemon keeps track of processes residing on its machine and participating in group communication. This information is shared between the daemons, creating the lightweight process group membership. The benefits of this client-daemon architecture are significant:

- The membership algorithm is invoked only if there is a change in the daemons' membership. Otherwise, when a process joins or leaves a group, the Spread daemon sends a notification message to the other daemons. When this message is ordered, the daemons deliver a membership notification containing the new group membership to the members of the group.
- Order is maintained at the daemons' level and not on a group basis. Therefore, for multi-group systems, message ordering is more efficient in terms of latency and excessive messages. Moreover, message ordering across groups is trivial since only one global order at the daemons' level is maintained.
- Implementing open groups, where processes that are not members of a group can multicast messages to the group, is easily supported.
- Flow control is maintained at the daemons' level rather than at the level of the individual process group. This leads to better overall performance in multi-group systems.

Several performance and scalability evaluations of the Spread toolkit are included below. The tests were conducted on 20 Pentium III 850Mhz Linux computers connected by a 100Mbps Fast Ethernet network. Figure 1.2 presents the total order throughput achieved by Spread as a function of the size of the network (number of daemons, each running on a separate computer) and the size of the multicast messages. In this experiment, half of the participating daemons serve a single local process each that multicasts to a specific group. Each of the other daemons serves a single local process that is a member of that group. For configurations ranging from 2 to 20 computers and message size above 1Kbytes a throughput of 60-80Mbits is achieved with a slight degradation as the number of participating computers is increased. Figure 1.3 presents the same experiment focusing on small messages. It is interesting to note the performance dip for messages around 700Bytes that happens when messages can no longer be packed into one network packet. Similar but less pronounced fragmentation effects can also be noticed in Figure 1.2 for larger message sizes.

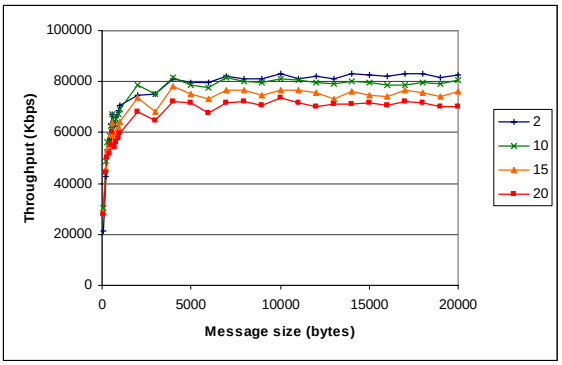

Figure 1.2 Throughput (large messages)

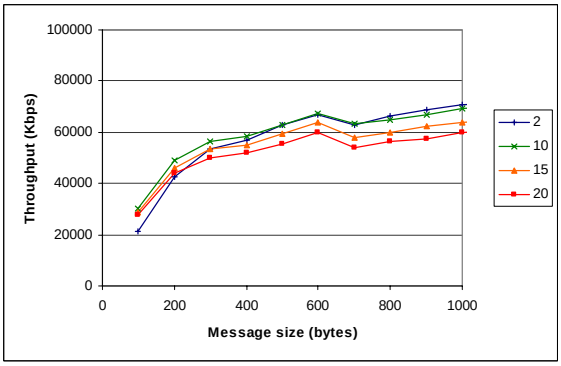

Figure 1.3 Throughput (small messages)

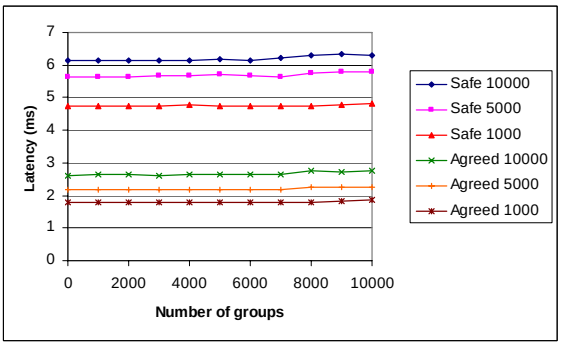

Figure 1.4 Message Latency

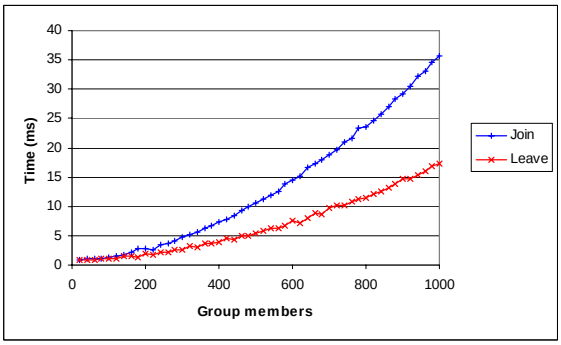

Figure 1.5 Lightweight Membership Latency

In a different experiment, we run Spread on 20 computers. On each computer a receiving application joins a certain number of groups, from 1 to 10,000. All the receiving applications on the different computers join the same set of groups. On one of the computers, a test application sends messages at a constant rate of 500Kbps, each message to a different group joined by the receiving applications. The one-way latency of each message received is recorded by the receiving applications. The clocks of the computers are accurately synchronized through a separate process similar to NTP. Figure 1.4 presents the message latency as a function of the number of groups joined by the receiving applications, the size of the multicast messages, and the type of the service (Agreed delivery for total order, or Safe delivery for stability). The latency of Agreed delivery (total order) ranges between 1.7ms to 2.8ms depending on the size of the message (1000, 5000 and 10000 bytes). The latency for Safe delivery (stability) ranges between 4.7ms to 6.4ms. The higher latency incurred by larger messages is mostly attributed to the time it takes to send them on a 100Mbits network. The figure shows that the number of groups in the system does not affect the message latency much. This is achieved thanks to a skip list data structure that provides log(n) access to the lightweight group structures in Spread.

In the last experiment, we run Spread on 20 computers. On each computer we run between 1 and 50 instances of a test application, each joining the same group. Therefore, in the group communication system as a whole there are between 20 to 1000 participating processes. A separate application joins and leaves the same group 100 times and the latency of each join and leave operation is measured. Figure 1.5 presents the average latency of the join and leave operations as a function of the group size. This experiment shows that joining a group that has 1000 members takes less than 40ms (including membership notifications to all 1000 members). As the number of members in the group increases, the size of the membership notification increases linearly (each member contributes about 32 bytes to the size of the notification), and the number of notifications per daemon also increases linearly. This explains the quadratic shape of the graph. The scalability with the number of groups and number of group participants in the system is attributed to the client-daemon architecture of Spread.