Algorithm 4 takes into consideration that machine allocation is symmetric for scale-in and scale-out. The important distinction between the starting and ending cluster sizes, therefore, is not before/after but larger/smaller. And the delta between the larger and smaller clusters is equal to the number of machines receiving data from the smaller cluster when scaling out, or the number of machines sending data to the smaller cluster when scaling in. These values are assigned to l, s and Δ in Lines 2 to 4 of Algorithm 4. Line 5 assigns to r the remainder of dividing Δ by s, which will be important later in the algorithm.

Given these definitions, the algorithm considers the three cases discussed in Section 4.4.1. In the first case, the size of the smaller cluster is greater than or equal to Δ, which means that all new machines must be allocated (or de-allocated) at once in order to allow for maximum parallel movement (Line 6). In the second case, Δ is a perfect multiple of the smaller cluster, so blocks of s machines will be allocated (or deallocated) at once and simultaneously filled (or emptied). Thus, the average number of machines allocated is $(2s+l)/2$ (Line 7). In the third case we have three phases, and the calculation of the average number of machines is shown in Lines 8 to 18 of Algorithm 4.

C B2W BENCHMARK

This appendix provides more detail about the B2W Benchmark introduced in Section 7. The transaction logs from B2W include the timestamp and the type of each transaction (e.g., GET, PUT, DELETE), as well as unique identifiers for the shopping carts, checkouts and stock items that were accessed or modified. Since there is some important information not available in the logs (e.g., the contents of each shopping cart), we also use a dump of all of the B2W shopping carts, checkouts, and stock data from the last year. The data has been anonymized to eliminate any sensitive customer information, but otherwise it is identical to the data in production. Joining the unique identifiers from the log data with the keys in the database dump thus allows us to infer almost everything about each transaction, meaning we can effectively replay the transactions starting from any point in the logs. This allows us to run H-Store with the same workload running in B2W's production shopping cart, checkout and stock databases.

A simplified database is shown in Figure 14, and a list of the transactions is shown in Table 4. When a customer tries to add an item to their cart through the website, GetStockQuantity is called to see if the item is available, and if so, AddLineToCart is called to update the shopping cart. At checkout time, the system attempts to reserve each item in the cart, calling ReserveStock on each item. If a given item is no longer available, it is removed from the shopping cart and the customer is notified. The customer has a chance to review the final shopping cart before they agree to the purchase.

Although the data used in this work is proprietary to B2W, the H-Store benchmark containing the full database schema and transaction logic is not. The benchmark is open-source and available on GitHub for the community to use [30].

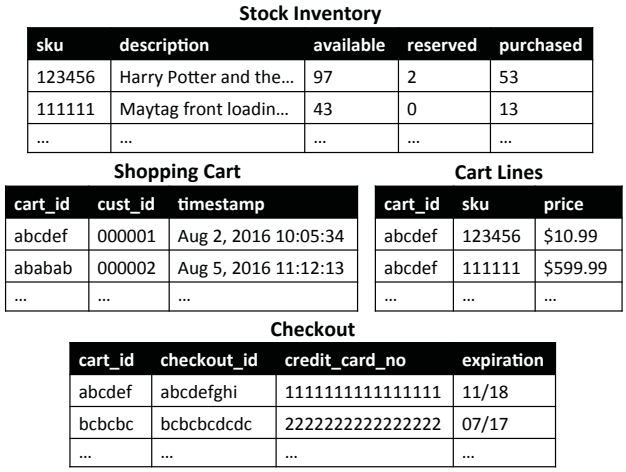

Figure 14: Simplified database for the B2W H-Store benchmark

Transaction	Description
AddLineToCart	Add a new item to the shopping cart, create the cart if it doesn't exist yet
DeleteLineFromCart	Remove an item from the cart
GetCart	Retrieve items currently in the cart
DeleteCart	Delete the shopping cart
GetStock	Retrieve the stock inventory information
GetStockQuantity	Determine availability of an item
ReserveStock	Update the stock inventory to mark an item as reserved
PurchaseStock	Update the stock inventory to mark an item as purchased
CancelStockReservation	Cancel the stock reservation to make an item available again
CreateStockTransaction	Create a stock transaction indicating that an item in the cart has been reserved
ReserveCart	Mark the items in the shopping cart as reserved
GetStockTransaction	Retrieve the stock transaction
UpdateStockTransaction	Change the status of a stock transaction to mark it as purchased or cancelled
CreateCheckout	Start the checkout process
CreateCheckoutPayment	Add payment information to the checkout
AddLineToCheckout	Add a new item to the checkout object
DeleteLineFromCheckout	Remove an item from the checkout object
GetCheckout	Retrieve the checkout object
DeleteCheckout	Delete the checkout object

Table 4: Operations from the B2W H-Store benchmark