VoltDB Do's & Don'ts

The following is a collection of best practices to make your work with VoltDB successful.

Do's

1. Partition your tables to maximize the frequency of single-partition transactions and minimize multi-partition transactions.
2. Estimate the data volumes and frequency of each transaction for your application to determine what columns to use for partitioning.
3. Be sure to indicate when a stored procedure is single-partitioned using the @ProcInfo annotation. (Otherwise VoltDB assumes it is multi-partitioned.)
4. Use multiple SQL queries in your stored procedures. Ten SQL queries in one single-partition procedure can be 10x faster than ten procedures with one query each.
5. Set the IsFinal flag to true on the last voltExecuteSQL() call in each stored procedure. This improves performance, especially for multi-partitioned transactions.
6. Benchmark, benchmark, benchmark. Different applications and hardware have different performance characteristics. Use benchmarking to determine the optimal number of partitions for your application.
7. Use asynchronous procedure calls and create client connections to all nodes in your cluster to maximize throughput.
8. If you encounter any confusing errors, get stuck, or need help, post your questions in the VoltDB forums (http://community.voltdb.com/forum) We are here to help.

Don'ts

1. Don't create tables with very large rows (that is, lots of columns or large VARCHAR columns). Several smaller tables with a common partitioning key are better.
2. Don't create queries that return large volumes of data (such as SELECT * FROM FOO with no constraints) especially for multi-partition transactions. Be conservative in the data returned by stored procedures.
3. Don't replicate a table just because it has a small number of rows. Even small tables must be partitioned if they are updated frequently.
4. Don't do extensive processing in your asynchronous callbacks. Only one callback is processed at a time, so make the callback procedures efficient to avoid stalls.
5. Don't try to benchmark your application on a single machine. Be sure to benchmark the impact of different cluster configurations.
6. Don't assume exceptionally low latency for any single VoltDB transaction. VoltDB is optimized for application throughput, not individual transaction latency. However, VoltDB's latency is competitive with other database products.
7. Don't call ClientFactory.createClient() more than once in each client application. There can be multiple client applications and each can have multiple connections to the database cluster. But there should be only one client instance within each client application.