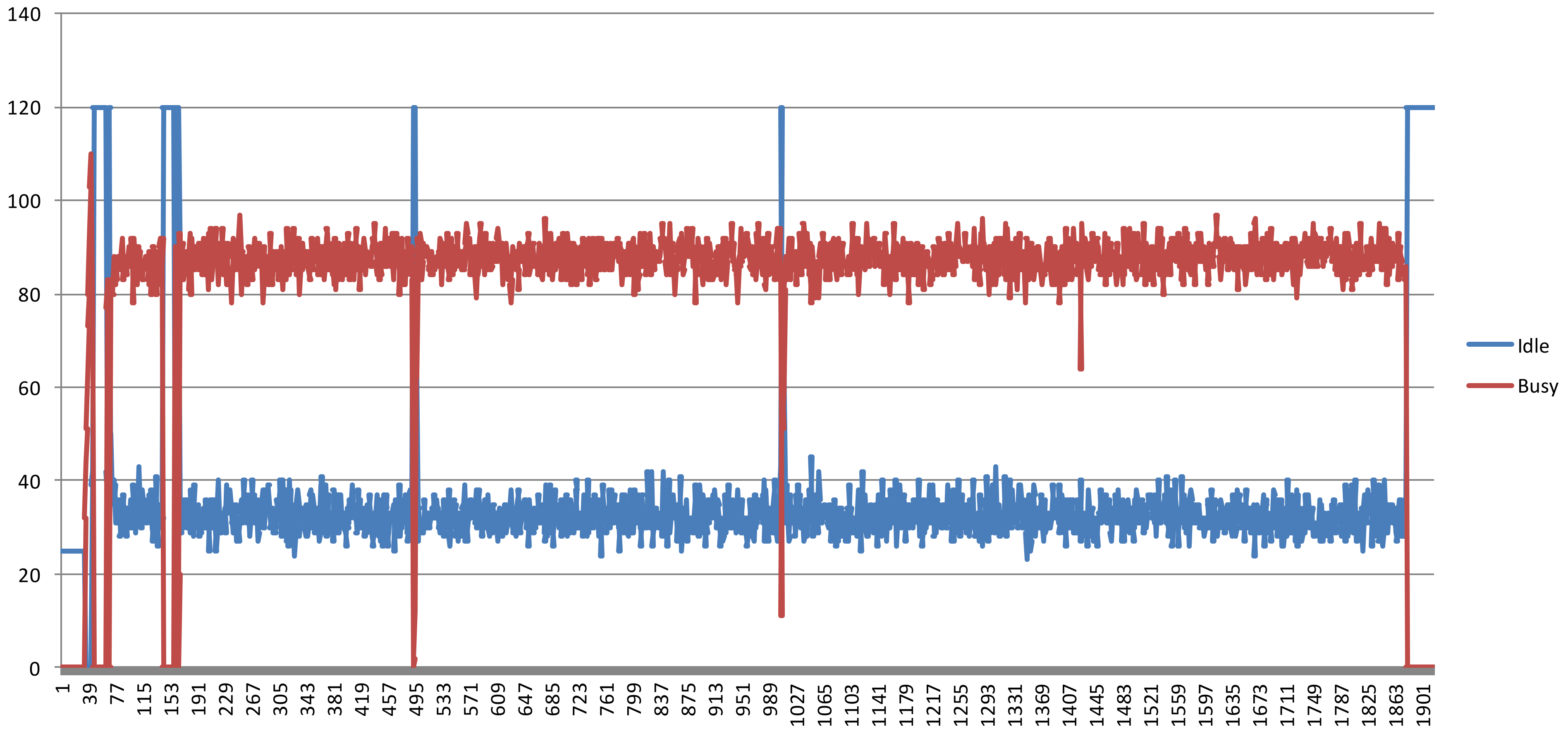
Setup Detail

- PoC 500 Card
- 32 Processor
- 132 GB RAM
- Get Directory 400 msg/sec
- Node.js Client [4 instance, each pumping 100 msg/sec]
- 1 hr traffic

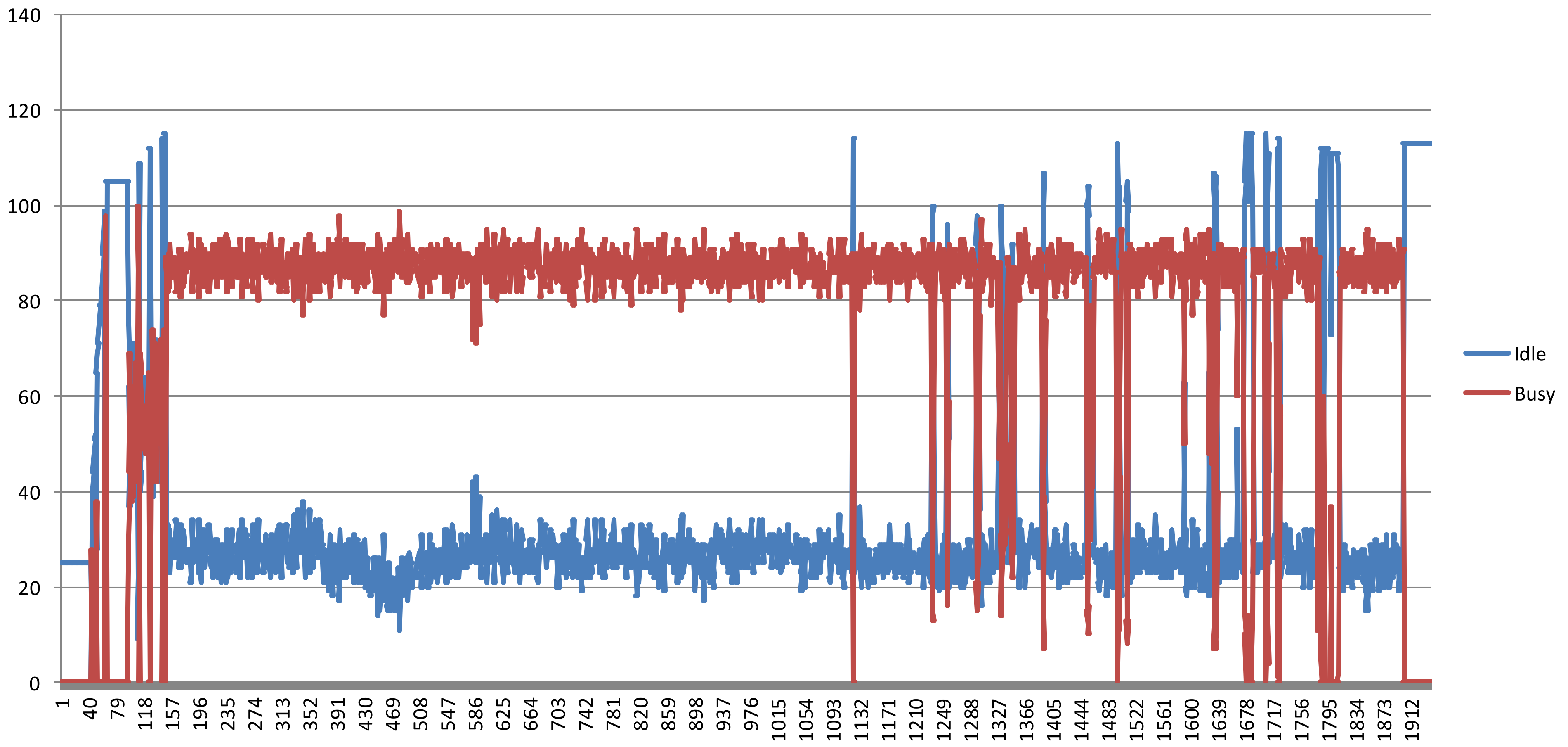
Latency

	Total Request	Average Latency	Maximum Latency	Failure
C3p0	1427995	0.25035042	18.947	75
Hikari	1424529	✓0.24209265	✓1.43	✓0
Tomcat Jdbc	1359448	0.25102726	19.056	✓0

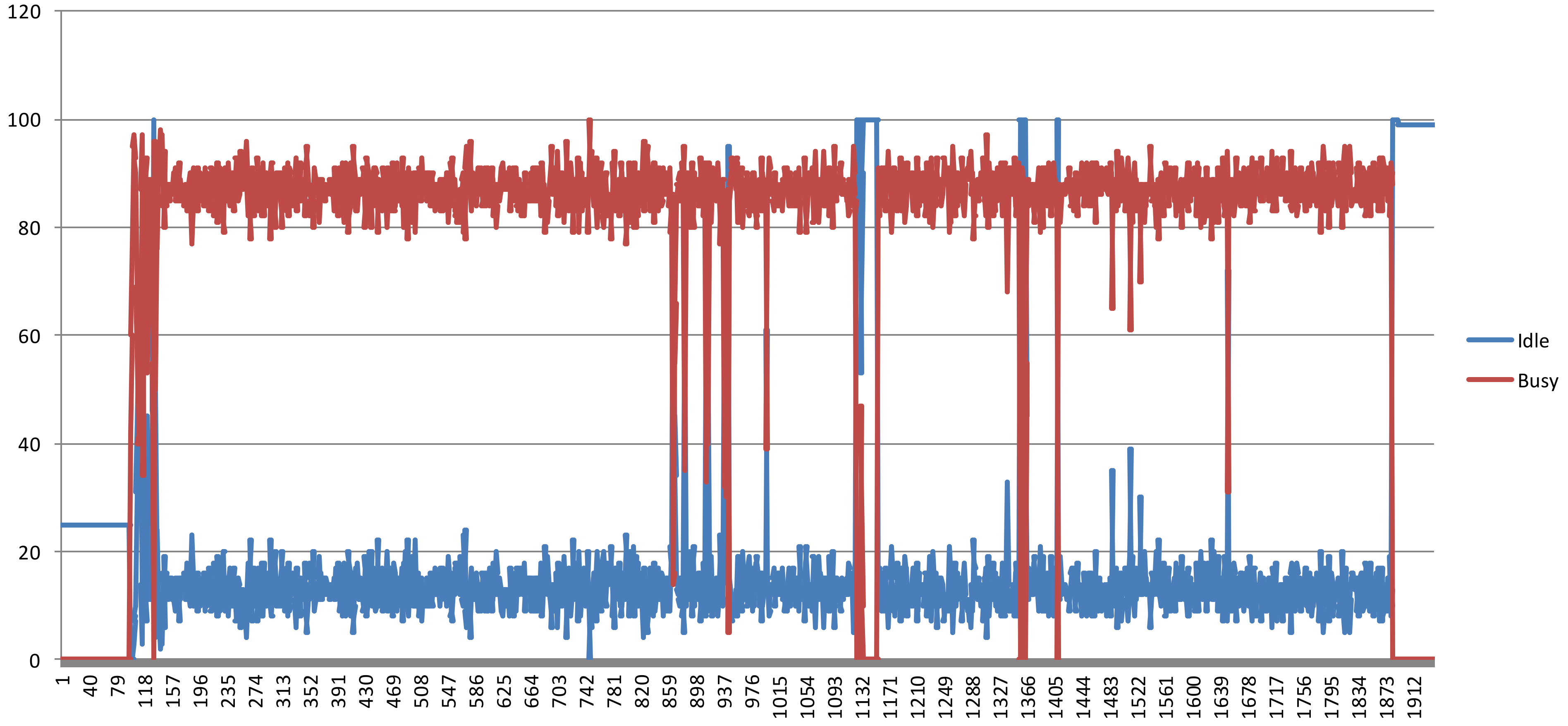
C3p0[Idle/Busy]



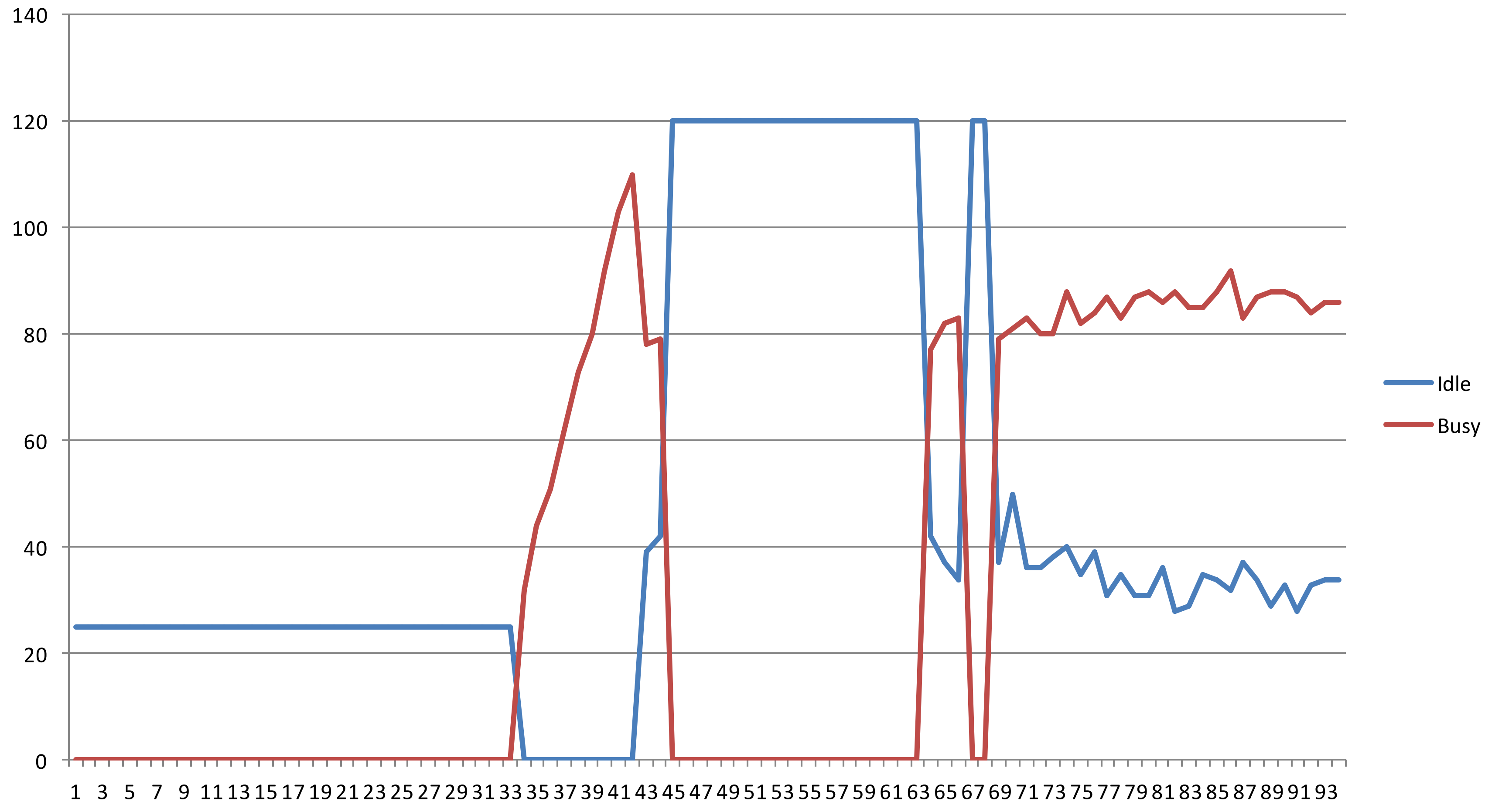
Hikari[Idle/Busy]



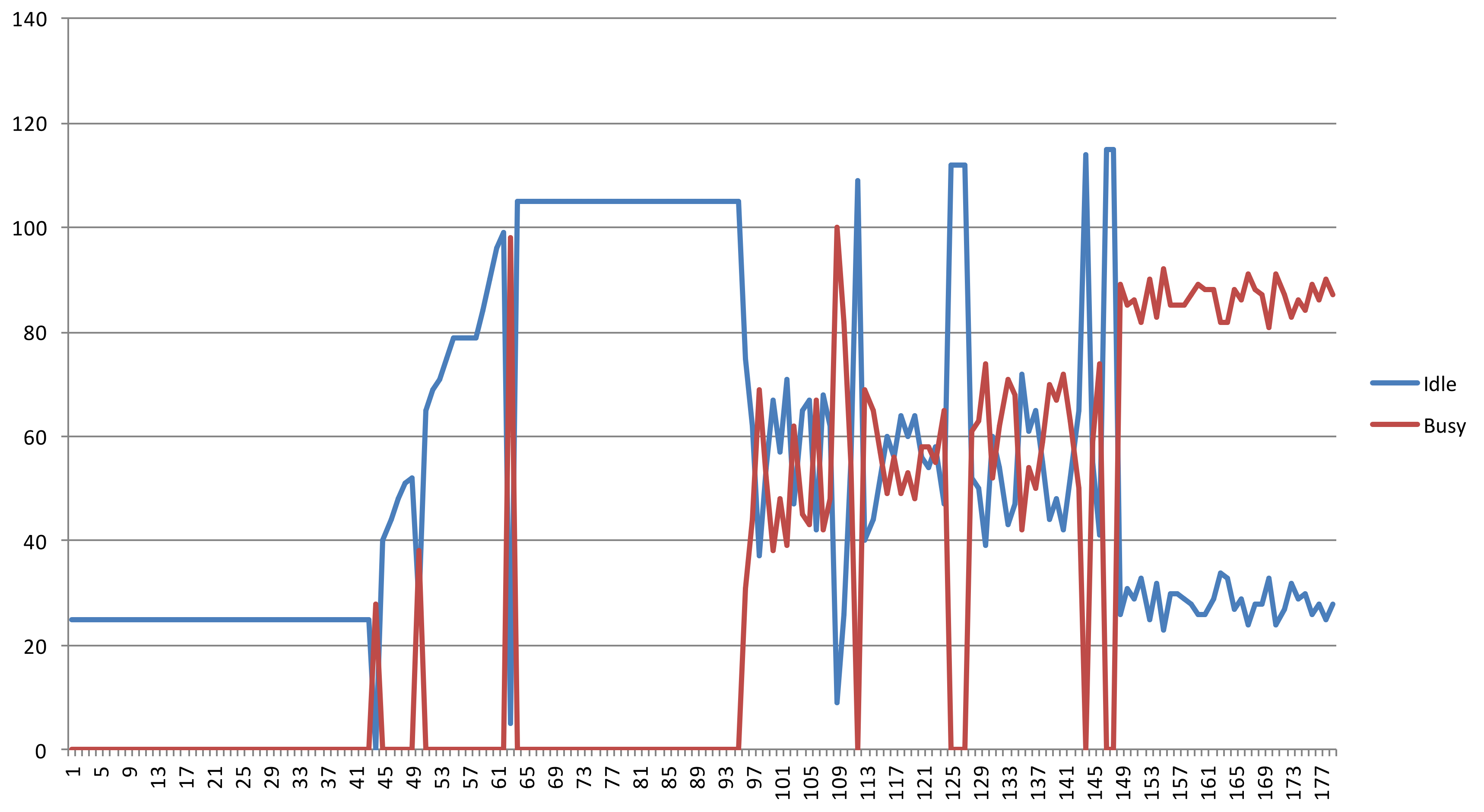
Tomcat jdbc pool[Idle/Busy]



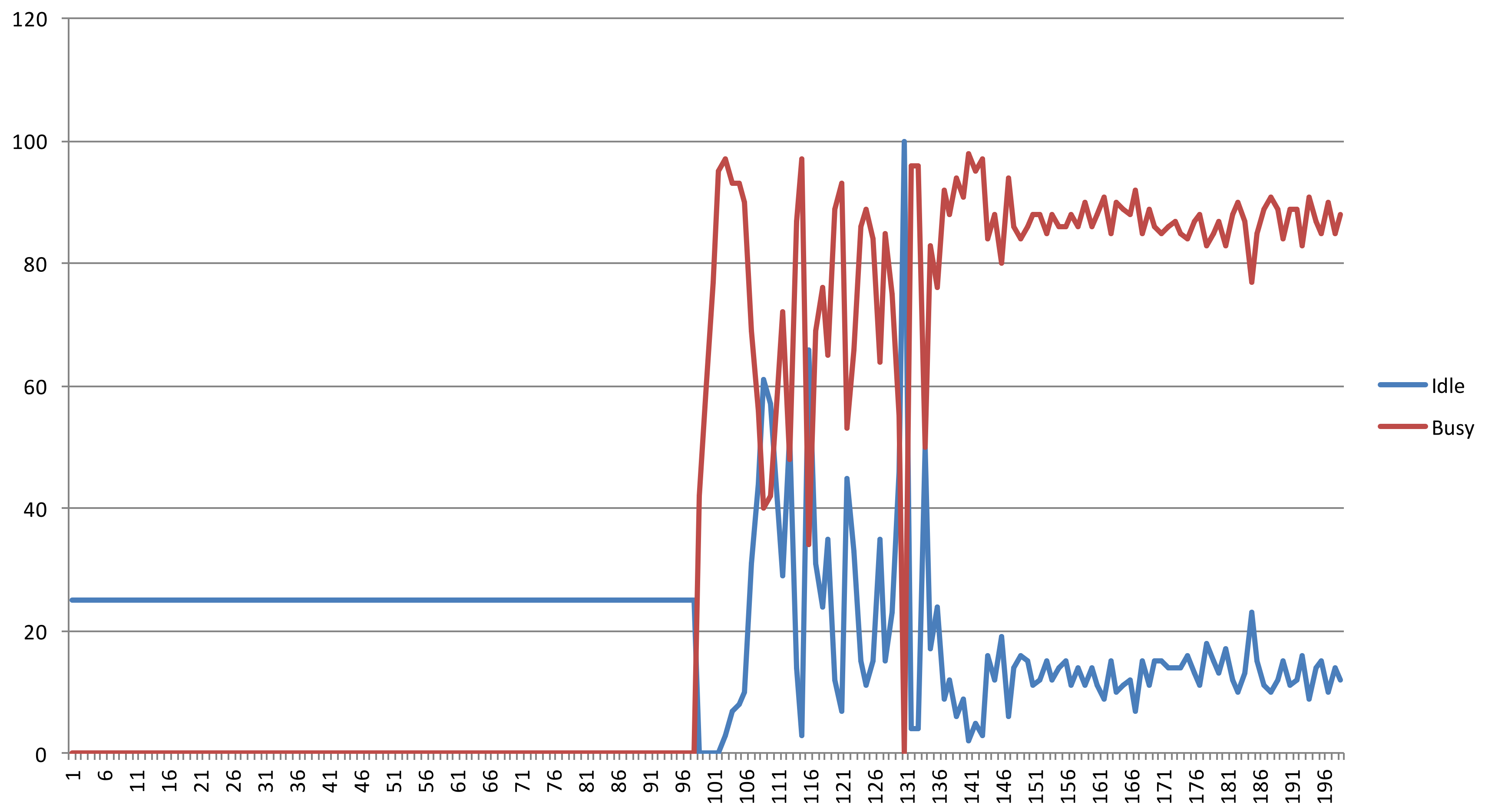
C3p0 Idle/Busy burst Graph



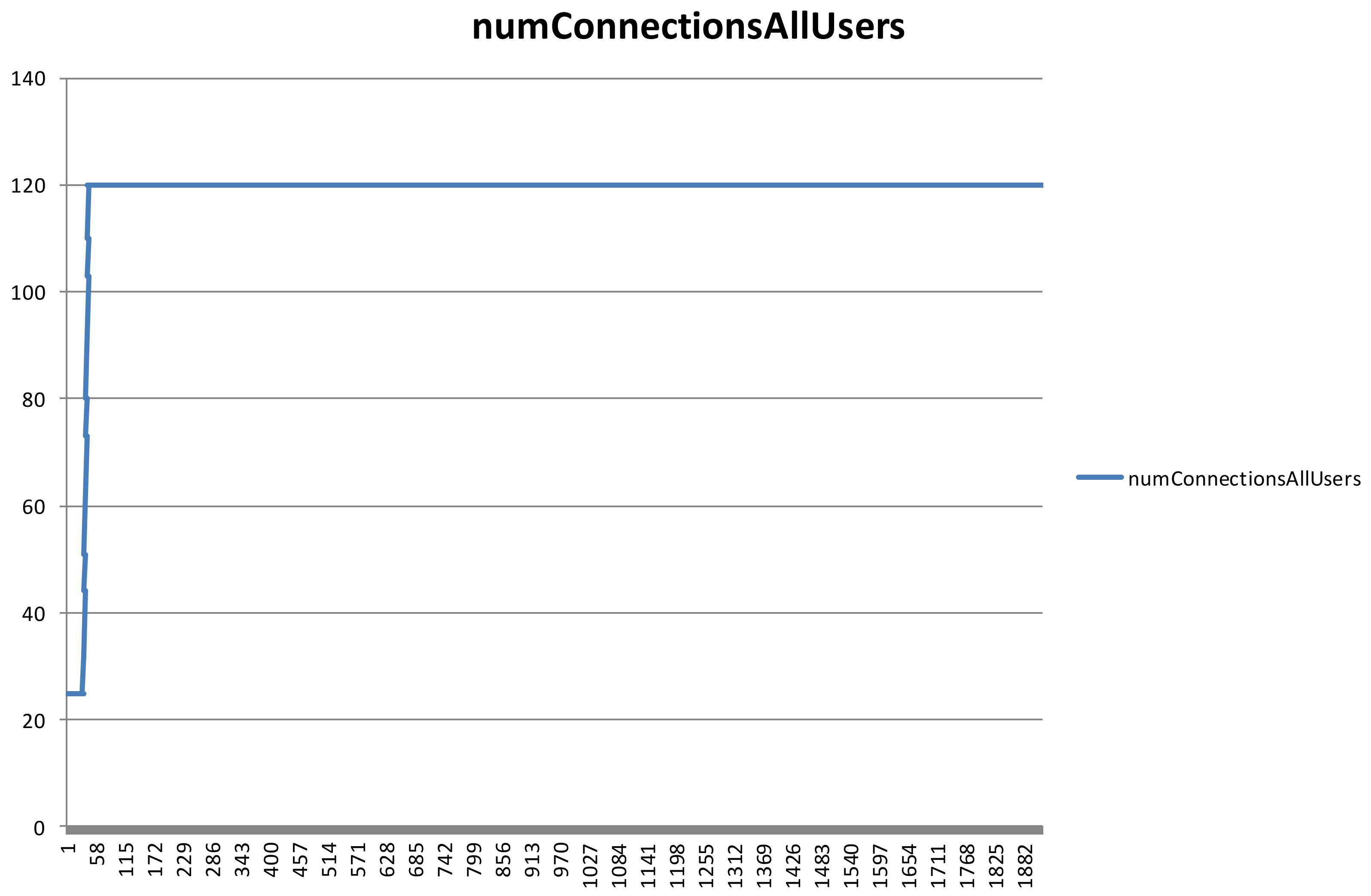
Hikari Idle/Busy burst Graph



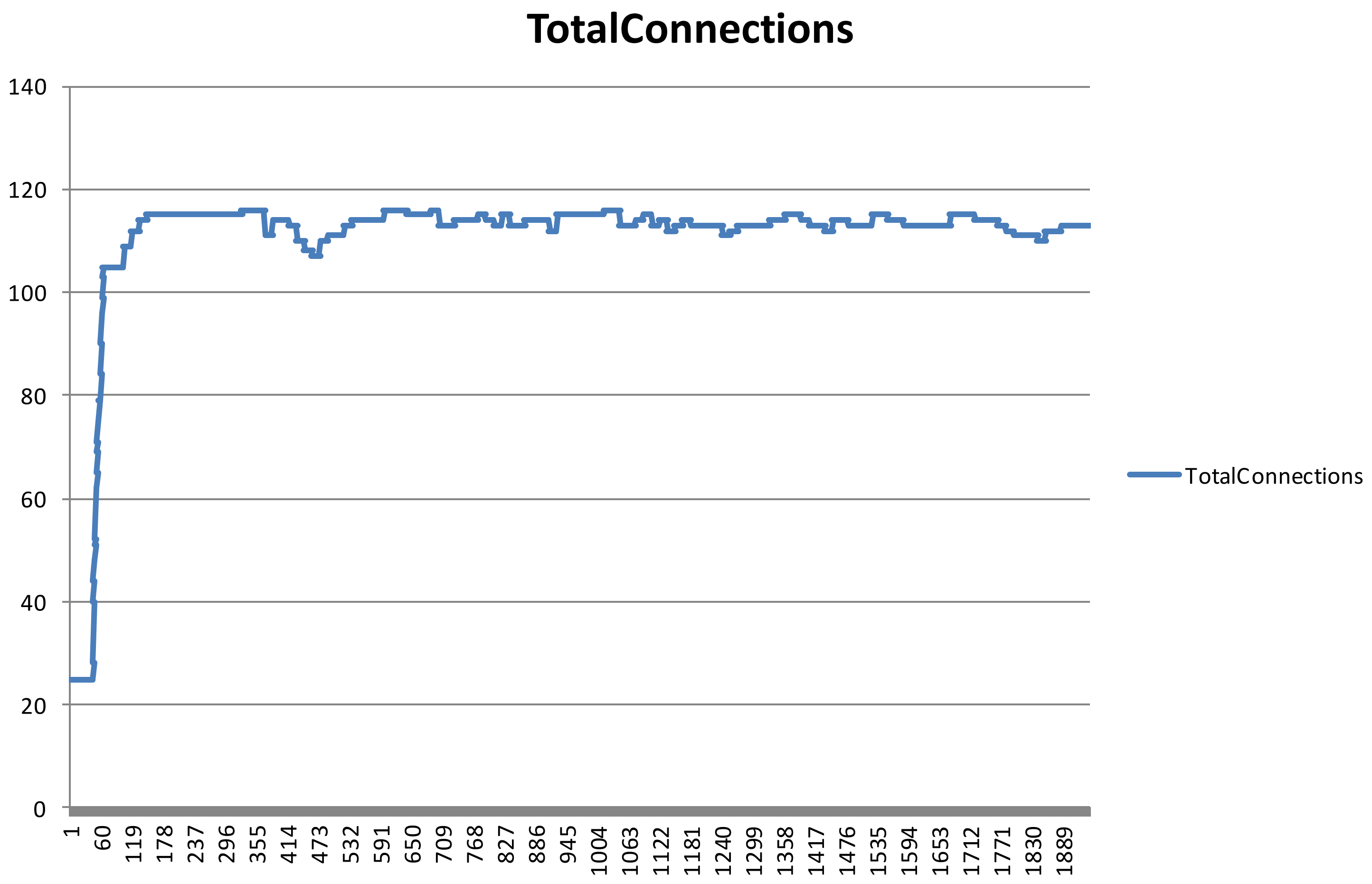
Tomcat Idle/Busy burst Graph



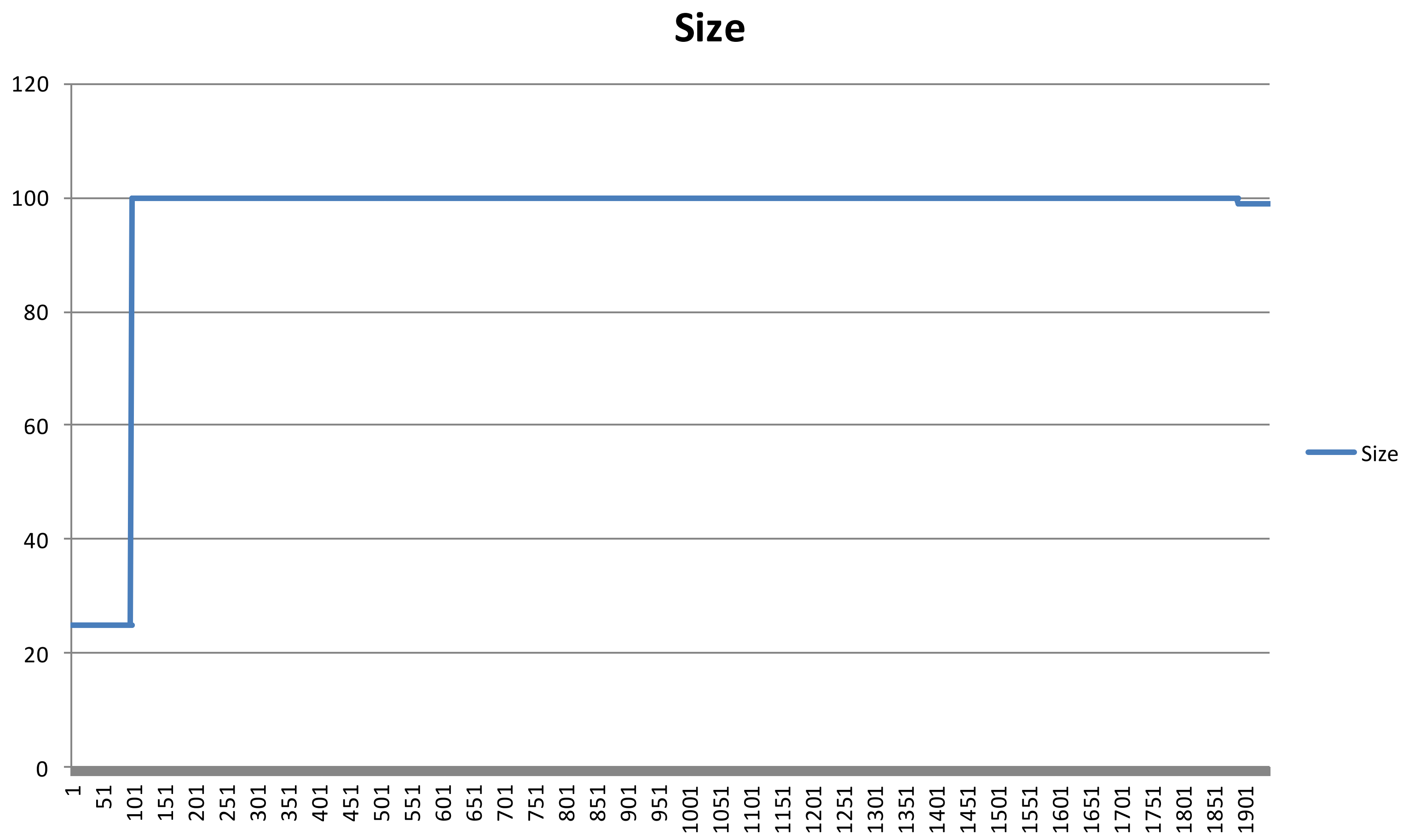
Total Connection c3p0



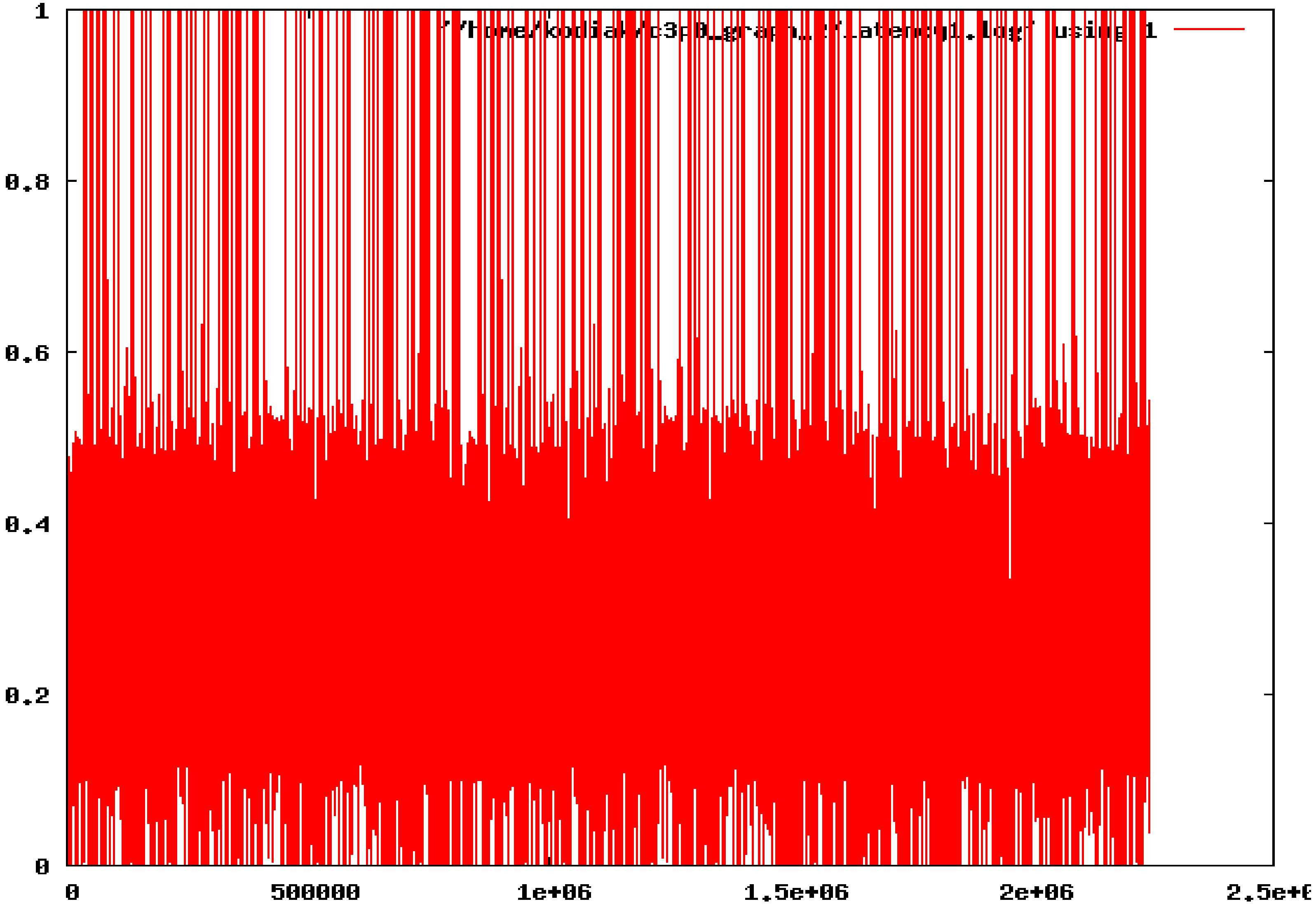
Total connection hikari



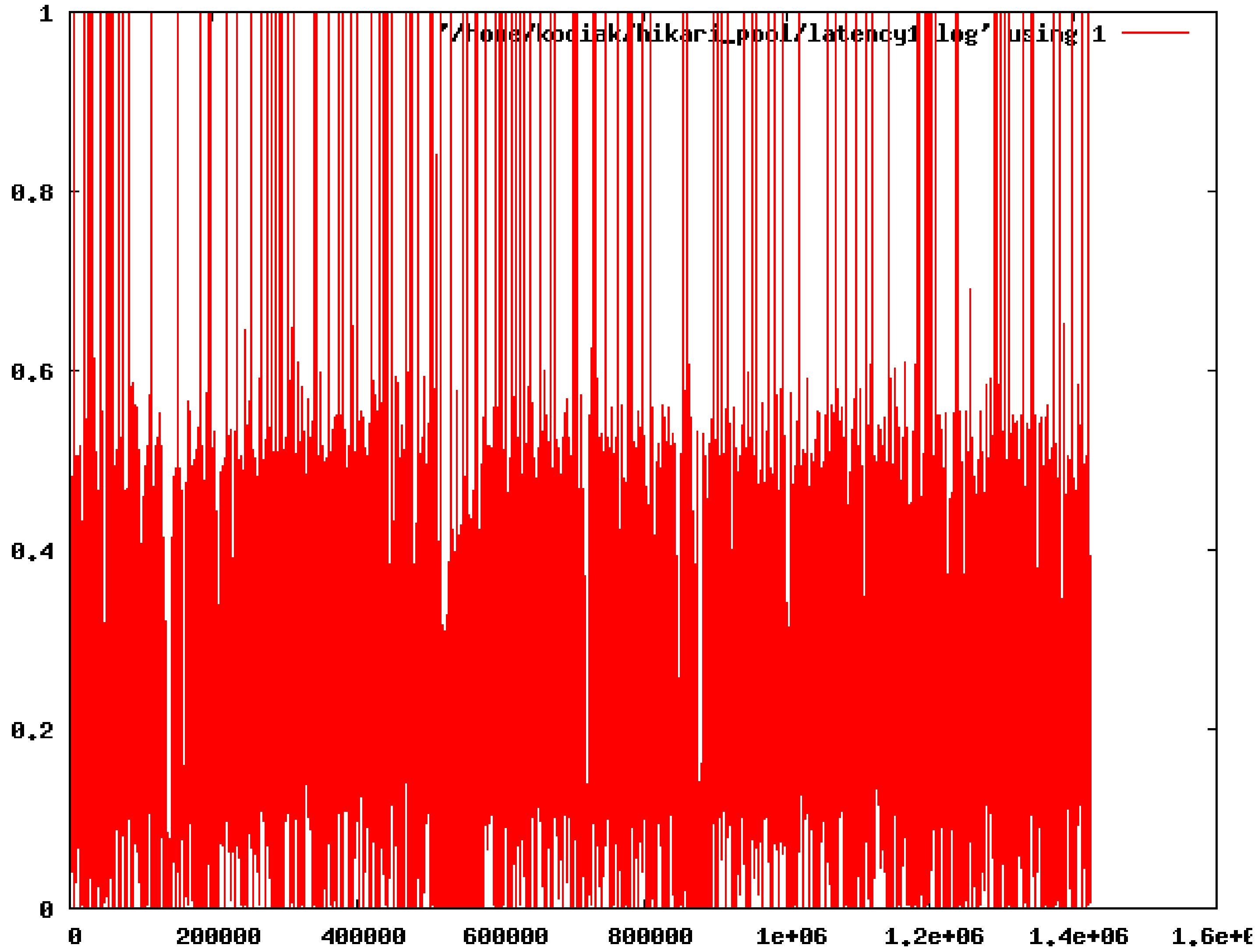
Total Connection tomcat



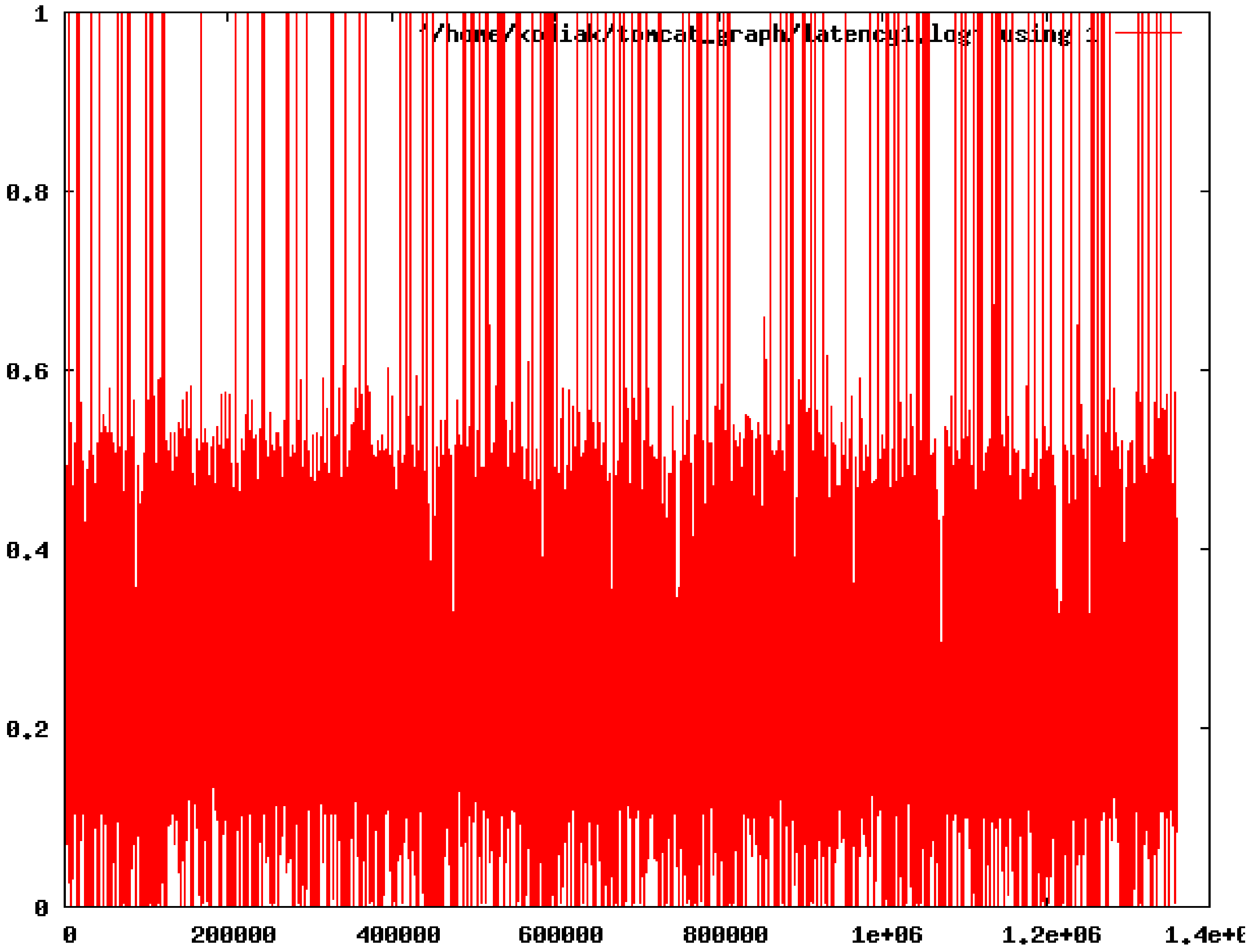
C3p0 latency



Hikari latency



Tomcat pool latency



C3p0 Connection Pool

acquireIncrement	factoryClassLocation	numHelperThreads
acquireRetryAttempts	forceIgnoreUnresolvedTransactions	overrideDefaultUser
acquireRetryDelay	forceUseNamedDriverClass	overrideDefaultPassword
autoCommitOnClose	idleConnectionTestPeriod	password
automaticTestTable	initialPoolSize	preferredTestQuery
breakAfterAcquireFailure	jdbcUrl	privilegeSpawnedThreads
checkoutTimeout	maxAdministrativeTaskTime	propertyCycle
connectionCustomizerClassName	maxConnectionAge	statementCacheNumDeferredCloseThreads
connectionTesterClassName	maxIdleTime	testConnectionOnCheckin
contextClassLoaderSource	maxIdleTimeExcessConnections	testConnectionOnCheckout
dataSourceName	maxPoolSize	unreturnedConnectionTimeout
debugUnreturnedConnectionStackTraces	maxStatements	user
driverClass	maxStatementsPerConnection	usesTraditionalReflectiveProxies
extensions	minPoolSize	

Hikari Connection Pool

dataSourceClassName	connectionTestQuery	registerMbeans
jdbcUrl	minimumIdle	catalog
username	maximumPoolSize	connectionInitSql
password	metricRegistry	connectionCustomizerClass Name
autoCommit	poolName	driverClassName
connectionTimeout	initializationFailFast	transactionIsolation
idleTimeout	isolateInternalQueries	leakDetectionThreshold
maxLifetime	readOnly	dataSource

Tomcat Connection Pool

factory	testWhileIdle	maxIdle
type	validationQuery	minIdle
defaultAutoCommit	validatorClassName	initialSize
defaultReadOnly	timeBetweenEvictionRunsMillis	maxWait
defaultTransactionIsolation	minEvictableIdleTimeMillis	testOnBorrow
defaultCatalog	removeAbandoned	testOnReturn
driverClassName	removeAbandonedTimeout	alternateUsernameAllowed
username	logAbandoned	dataSourceJNDI
password	connectionProperties	validationInterval
maxActive	initSQL	jmxEnabled
abandonWhenPercentageFull	jdbcInterceptors	fairQueue
maxAge	useEquals	suspectTimeout

getDirectory Latency using c3p0[400msg/sec,Remote Logging]

- Total Request=701632
- Total Latency=6423.539
- Average Latency=**0.00915514 [with local logging its around 250 ms]**
- Min Latency=0.001 Cid=36799361
- Max Latency=15.165 Cid=36774379

getDirectory Latency using Tomcat JDBC [400msg/sec,Remote Logging]

- Total Request=713290
- Total Latency=8411.114
- Average Latency=**0.011791998 [with local logging its around 250 ms]**
- Min Latency=0.001 Cid=36071999
- Max Latency=16.336 Cid=36053954

Conclusion

Tomcat JDBC Pool benefit against c3p0:-

- **No Latency improvement**
- Using **100 connection** instead of 120 connection in getDirectory 400 msg/sec
- No failure in burst traffic [burst of 100 message], c3p0 on an average of **70 failure** for same burst of 100 message rate

Conclusion

- Local logging is the bottleneck
- With Remote logging 99.98% improvement is there in latency of getDirectory 400 msg/sec
- If we will not consider burst traffic we can stay with c3p0 connection pooling instead of switching to tomcat jdbc pool.
- Connection pool is not the bottleneck