Proposal: Asynchronous Sending for RemoteSyslogAppender in log4net

Summary:

This document proposes an enhancement to the RemoteSyslogAppender in Apache log4net to resolve severe performance degradation observed under high load conditions due to synchronous network I/O.

Problem Statement:

When using RemoteSyslogAppender in log4net version **3.1.0.0**, we observed **significant performance degradation** under load, particularly during high throughput API requests in a production-scale ASP.NET application.

Symptoms:

- API response time increases drastically with RemoteSyslogAppender enabled.
- When the appender is disabled, APIs perform consistently faster.
- The bottleneck was traced to the following line in the Append() method:

  Client.SendAsync(buffer, buffer.Length, RemoteEndPoint).Wait();

Root Cause:

- This line blocks the thread by waiting on an **asynchronous UDP send** operation.
- In high-load scenarios, this creates **thread contention**, as logging becomes I/O-bound and synchronous.
- UDP is inherently unreliable and fast; blocking to wait for its completion defeats the purpose and affects overall throughput.

Proposed Solution:

Refactor the RemoteSyslogAppender to **decouple** the UDP send operation from the calling thread by using a **producer-consumer pattern** backed by BlockingCollection<byte[]>.

Key Enhancements:

1. **Asynchronous Background Worker**: A background task consumes queued log messages and sends them via UDP.
2. **Non-blocking Logging Path**: Append() only queues the message—does not wait for UDP transmission.

3. **Graceful Shutdown**: Ensures buffered logs are flushed before shutdown.

Code Changes:

```
private readonly BlockingCollection<byte[]> _sendQueue = new();
private CancellationTokenSource? _cts;
private Task? _pumpTask;

public override void ActivateOptions()
{
   base.ActivateOptions();
   _cts = new CancellationTokenSource();
   _pumpTask = Task.Run(() => ProcessQueueAsync(_cts.Token), CancellationToken.None);
}

protected override void OnClose()
{
   _cts?.Cancel();
   _pumpTask?.Wait(TimeSpan.FromSeconds(5));
   base.OnClose();
}

protected override void Append(LoggingEvent loggingEvent)
{
   var buffer = FormatMessage(loggingEvent); // Assuming your existing logic
   _sendQueue.Add(buffer);
}

private async Task ProcessQueueAsync(CancellationToken token)
{
   using (var udp = new UdpClient())
```

```
    {
        udp.Connect(RemoteAddress?.ToString(), RemotePort);

        try
        {
            while (!token.IsCancellationRequested)
            {
                var datagram = _sendQueue.Take(token);
                try
                {
                    await udp.SendAsync(datagram, datagram.Length);
                }
                catch (Exception ex) when (!ex.IsFatal())
                {
                    ErrorHandler.Error("RemoteSyslogAppender: send failed", ex,
ErrorCode.WriteFailure);
                }
            }
        }
        catch (OperationCanceledException)
        {
            while (_sendQueue.TryTake(out var leftover))
            {
                try { await udp.SendAsync(leftover, leftover.Length); } catch { }
            }
        }
    }
}
```

Technical Justification:

- BlockingCollection<T> with a dedicated task ensures **high throughput** and **thread safety**.
- UdpClient.SendAsync() is naturally asynchronous and works well in an async context.
- This pattern avoids .Wait() and prevents **thread pool starvation** under high loads.
- Maintains the **contract** of RemoteSyslogAppender without changing external behavior.

Performance Evidence:

Environment:

- .NET framework 4.7.2 application
- log4net 3.1.0.0
- Remote syslog server running on a separate host
- Load test using Apache JMeter with 200 concurrent users

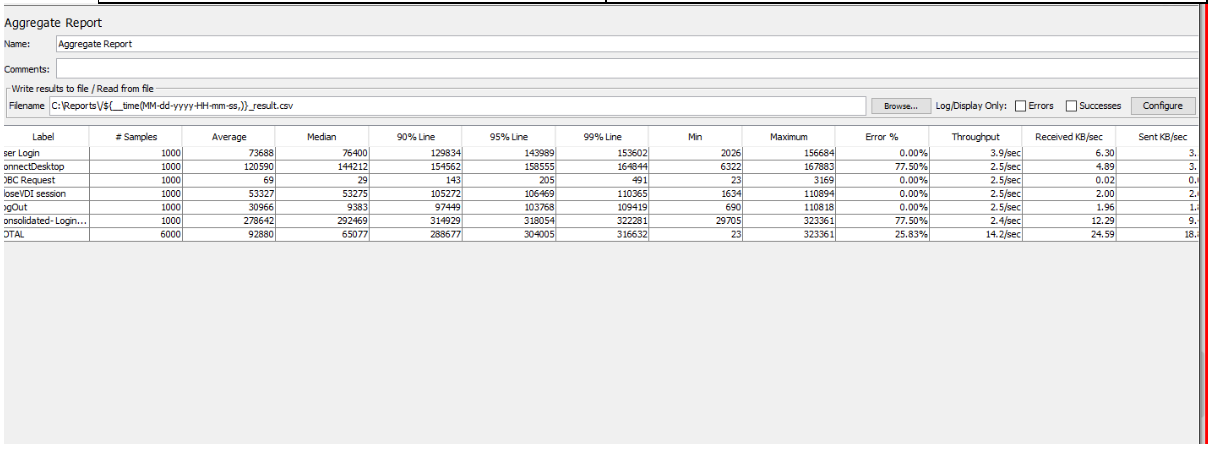
Test Configuration

- Tool: Apache JMeter
- API Endpoint: High-throughput production endpoint
- Threads: 1000
- Ramp-Up Period: 100 seconds
- Request Rate: 10 requests/second

Scenario 1: **Default log4net with RemoteSyslogAppender Enabled (Synchronous.Wait())**

Metric	**Value**
Average Response Time	120590
Minimum Response Time	6322
Maximum Response Time	167883

Errors/Failures	Some API errors due to timeouts or long waits



Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename C:\Reports\/${__time(MM-dd-yyyy-HH-mm-ss,)}_result.csv

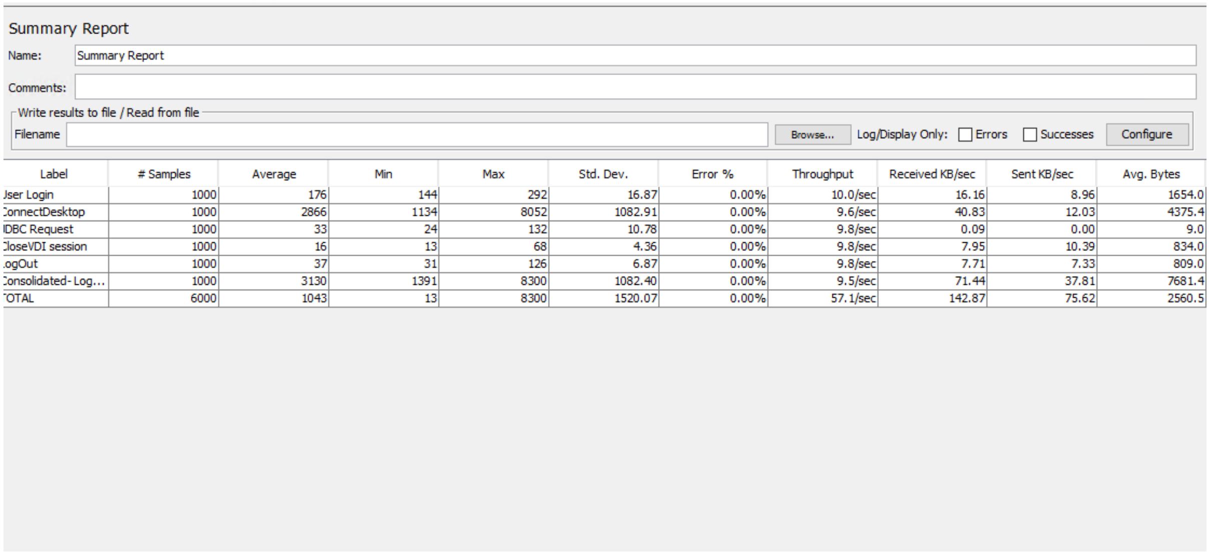
Log/Display Only: Errors Successes

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
ser Login	1000	73688	76400	129834	143989	153602	2026	156684	0.00%	3.9/sec	6.30	3.
onnectDesktop	1000	120590	144212	154562	158555	164844	6322	167883	77.50%	2.5/sec	4.89	3.
⊃BC Request	1000	69	29	143	205	491	23	3169	0.00%	2.5/sec	0.02	0.
loseVDI session	1000	53327	53275	105272	106469	110365	1634	110894	0.00%	2.5/sec	2.00	2.
ɔgOut	1000	30966	9383	97449	103768	109419	690	110818	0.00%	2.5/sec	1.96	1.
onsolidated- Login…	1000	278642	292469	314929	318054	322281	29705	323361	77.50%	2.4/sec	12.29	9.
⊃TAL	6000	92880	65077	288677	304005	316632	23	323361	25.83%	14.2/sec	24.59	18.

Issues Observed: RemoteSyslogAppender causes the API thread to block due to .Wait(), resulting in slow responses.

Scenario 2: log4net with Asynchronous RemoteSyslogAppender (Proposed Change)

Metric (for major api ConnectDesktop call)	Value
Average Response Time	2866
Minimum Response Time	1134
Maximum Response Time	8052
Errors/Failures	Some API errors due to long waits



Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename

Log/Display Only: Errors Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Jser Login	1000	176	144	292	16.87	0.00%	10.0/sec	16.16	8.96	1654.0
ConnectDesktop	1000	2866	1134	8052	1082.91	0.00%	9.6/sec	40.83	12.03	4375.4
IDBC Request	1000	33	24	132	10.78	0.00%	9.8/sec	0.09	0.00	9.0
CloseVDI session	1000	16	13	68	4.36	0.00%	9.8/sec	7.95	10.39	834.0
LogOut	1000	37	31	126	6.87	0.00%	9.8/sec	7.71	7.33	809.0
Consolidated- Log…	1000	3130	1391	8300	1082.40	0.00%	9.5/sec	71.44	37.81	7681.4
TOTAL	6000	1043	13	8300	1520.07	0.00%	57.1/sec	142.87	75.62	2560.5

Observation:

Avg, min max time taken is improved and less with With Async Queueing 10req/sec (thread 1000, ramp up 100sec) as compare to results with the Default log4net (sync .Wait())

Summary:

The proposed update improves the performance and responsiveness of applications using RemoteSyslogAppender, especially under load. By adopting an asynchronous, queue-based architecture, we eliminate the unnecessary blocking behavior caused by .Wait(), aligning better with modern asynchronous .NET patterns.

Next Steps:

We respectfully request the following actions from the log4net maintainers:

- **Review the proposed enhancement** for performance improvement in RemoteSyslogAppender.
- **Verify compatibility** with existing appender behavior.
- **Incorporate the fix** in the **next official release** of log4net.