Implementasi Web API pada Aplikasi Desktop untuk Meningkatkan Performa Aplikasi

Tony Wijaya
STMIK Pontianak
Jl. Merdeka No. 372 Pontianak, (0561) 735555
e-mail: tony_wijaya@stmikpontianak.ac.id

Abstrak

Penerapan connection pooling pada aplikasi desktop yang diteliti pada penelitian sebelumnya sudah terbukti dapat meningkatkan performa secara signifikan. Pada penelitian ini dilakukan implementasi web API pada aplikasi desktop supaya mencapai performa yang semakin baik lagi. Teknik connection pooling masih memiliki beberapa kelemahan baik di sisi client maupun di sisi server. Hal tersebut dapat diatasi dengan penerapan web API. Pengujian dilakukan dengan metode black box untuk membuktikan efisiensi antara penerapan teknik connection pooling dengan web API. Melalui penelitian ini diharapkan dapat membantu pengembang perangkat lunak dalam mengembangkan aplikasi desktop dengan database jarak jauh atau berbasis cloud dengan performa yang maksimal.

Kata kunci— Aplikasi desktop, database jarak jauh, web API.

Abstract

mplementation of connection pooling in desktop application from previous research proved to have increased app performance significantly. In this research, however, an even more increased performance in desktop application is gained through implementing web API. Connection pooling technique still has drawback either on client or server side. Implementation of web API can eliminate those drawbacks. Agile Methodology with Extreme Programming is used in this research. Black box testing is used to compare efficiency between connection pooling technique and web API. Hopefully this research can help software developers especially desktop application with remote or cloud-based database build their application with high performance.

Keywords— *Desktop application, remote database, web API.*

1. Pendahuluan

Penelitian terdahulu mengenai connection pooling terbukti berhasil meningkatkan performa aplikasi desktop yang mempunyai basis data jarak jauh [1]. Aplikasi desktop mengalami penurunan performa apabila dikoneksikan ke basis data di luar Local Area Network (LAN), seperti di Internet. Dengan connection pooling [2], koneksi di basis data dipertahankan untuk jangka waktu tertentu sehingga aplikasi desktop tidak perlu membuat koneksi baru setelah akses pertama kali. Koneksi pertama kali ke basis data memerlukan waktu yang relatif lama karena proses handshake [3] jarak jauh. Tanpa connection pooling [4], aplikasi harus melakukan proses tersebut berulang-ulang sehingga performa aplikasi menurun tajam.

Kelemahan dari teknik connection pooling terdapat pada akses pertama kali, karena proses handshake masih harus dilakukan secara jarak jauh. Proses ini rata-rata membutuhkan waktu 1-5 detik dan dirasakan oleh setiap komputer yang mengakses ke server. Jadi apabila terdapat 5 komputer, maka masing-masing komputer akan mengalami delay tersebut pada saat pertama kali mengakses aplikasi. Pooling koneksi juga tidak akan bertahan selamnanya. Sebuah pooling hanya bertahan hingga 10 menit. Apabila pemakai tidak mengakses basis data selama rentang waktu tersebut, maka koneksi akan dihancurkan di sisi server. Jika demikian, maka pemakai perlu menginisialisasi koneksi lagi dan mengalami delay yang sama seperti kali pertama mengakses aplikasi. Dilihat dari sisi server, teknik connection pooling juga membutuhkan sumber daya yang lebih besar. Apabila terdapat 5 komputer client yang mengakses basis data, maka server akan menciptakan 5 pool yang berbeda. Jadi sumber daya yang digunakan akan berbanding lurus dengan jumlah komputer client. Dengan kata lain, semakin banyak client, maka server akan menghabiskan semakin banyak sumber dayanya.

Web API atau REST ful web service merupakan sebuah arsitektur dalam metode pertukaran data web service [5]. Web API menggunakan data dalam format JavaScript Object Notation (JSON) dalam

proses pertukaran data antara client dan server. Apabila aplikasi desktop menggunakan web API sebagai middleware dalam koneksi ke basis data, maka aplikasi desktop tidak perlu lagi melakukan proses handshake ke server. Alih-alih menginisialisasi koneksi ke basis data jarak jauh, aplikasi dekstop mengakses web API yang menggunakan protokol standar HTTP. Web API inilah yang nantinya melakukan koneksi ke basis data. Hal ini berarti terjadi peningkatan performa yang signifikan pada aplikasi tersebut. Di samping itu, jumlah komputer client yang melakukan koneksi ke web API tidak berbanding lurus dengan sumber daya yang digunakan. Hal ini dikarenakan semua komputer client dikelola oleh 1 koneksi saja yaitu web API itu sendiri. Hal ini disamping tidak boros sumber daya, juga dapat digunakan sebagai router yang dapat mengantrekan setiap permintaan yang masuk ke basis data.

Di zaman peralihan ke cloud seperti sekarang ini, tidak heran apabila sebagian besar aplikasi sudah dibuat untuk arsitektur cloud pula. Hal ini juga berlaku bagi aplikasi desktop, baik yang sudah dibuat pada sistem lama maupun aplikasi desktop yang baru dibuat untuk sistem baru. Tidak asing lagi apabila aplikasi desktop pun mengelola basis data di cloud. Dengan mengkombinasikan aplikasi desktop dengan middleware berupa web API yang menjadi penengah antara aplikasi desktop dan basis data di cloud, akan tercapai performa yang sangat tinggi.

Objek penelitian ini adalah aplikasi SIU STMIK Pontianak. Aplikasi desktop ini digunakan oleh para staf akademik STMIK Pontianak untuk mengelola data Kartu Rencana Studi (KRS) mahasiswa, nilai mahasiswa, absensi, jadwal kuliah dan lain-lain.

Modul yang akan dibahas adalah modul distribusi mata kuliah yang berisi data dosen yang mengampuh mata kuliah untuk setiap kelasnya beserta jumlah SKS. Data ini dikelompokkan per semester dan tahun akademik. Penelitian ini akan membahas bagaimana:

- Aplikasi desktop mengakses ke web API di server online (cloud),
- Web API di server online (cloud) mengakses basis data
- Web API data dalam format JSON ke aplikasi desktop
- Aplikasi desktop menerima data JSON, mengolahnya dan menampilkan ke layar
- Hasil perbandingan kecepatan antara teknik connection pooling dengan web API

Penelitian ini ingin membuktikan bahwa web API mampu meningkatkan performa aplikasi desktop dibanding ketika aplikasi tersebut menggunakan teknik connection pooling sehingga pada akhirnya dapat meningkatkan produktivitas para staf akademik STMIK Pontianak..

2. Metode Penelitian

Penelitian ini dilakukan dengan pendekatan kualitatif [6]. Objek penelitian adalah aplikasi desktop SIU STMIK Pontianak. Instrumen penelitian yang digunakan dalam penelitian ini adalah:

- Postman versi 9.0.3 untuk menguji web API
- Web API “https://stmikpontianak.net/SiakStmikPtkApi/” yang merupakan web API untuk melayani request dari aplikasi desktop SIU STMIK Pontianak
- Aplikasi desktop SIU STMIK Pontianak khususnya modul “Distribusi Mata Kuliah”.

Kesimpulan dibuat berdasarkan interpretasi data oleh peneliti. Bentuk penelitian adalah penelitian eksperimen dengan objek yaitu aplikasi SIU STMIK Pontianak.

Metode perancangan perangkat lunak yang digunakan adalah metode Agile [7]. Prinsip dasar Agile tertuang dalam Agile Manifesto for Software Development [8]. Selain itu, metode ini juga dapat merespon dengan cepat terhadap perubahan daripada hanya mengikuti rancangan UML yang dibuat. Hal ini relevan karena penelitian ini dilakukan dengan bentuk eksperimen.

Pendekatan perancangan perangkat lunak yang digunakan adalah Extreme Programming. Extreme Programming merupakan disiplin dari pengembangan perangkat lunak yang berdasar pada nilai-nilai kesederhanaan, komunikasi, umpan balik, dan keberanian [9]. XP melibatkan semua anggota tim dalam setiap pekerjaannya: pair programming (coding bersama) dan semua pekerjaan lainnya secara bersama-sama. Jadi setiap anggota tim terlibat dalam semua aktifitas dan tidak ada pembagian tugas. Semua memiliki tanggung jawab yang sama, teknik menulis program yang sama, dan lain sebagainya. Hal ini sangat cocok diterapkan dalam penelitian ini karena coding dalam penelitian ini dilakukan oleh 1 orang saja. Karena dalam XP, setiap anggota tim juga melakukan semua pekerjaan tanpa ada pembagian tugas.

Figure 1. Sikuls Hidup Extreme Programming

1. Aplikasi SIU STMIK Pontianak dibangun di atas framework .NET [10] Windows Form [11] menggunakan bahasa pemrograman C# [12] dan dikode dengan bantuan IDE Visual Studio 2017 [13]. Server basis data STMIK Pontianak ada di cloud yang berupa Virtual Private Server (VPS) dan menggunakan sistem operasi Ubuntu 20.04 LTS (Long Time Support) [14]. Basis data yang digunakan adalah MariaDB 10.3.30 yang merupakan basis data yang bersifat open source [15]. MariaDB digunakan karena sangat mendukung dengan sistem operasi yang digunakan yaitu Ubuntu sehingga menghasilkan performa yang sangat tinggi.

3.HASIL DAN PEMBAHASAN

Tampilan modul "Distribusi Mata Kuliah" pada aplikasi SIU STMIK Pontianak dapat dilihat pada gambar di bawah ini:

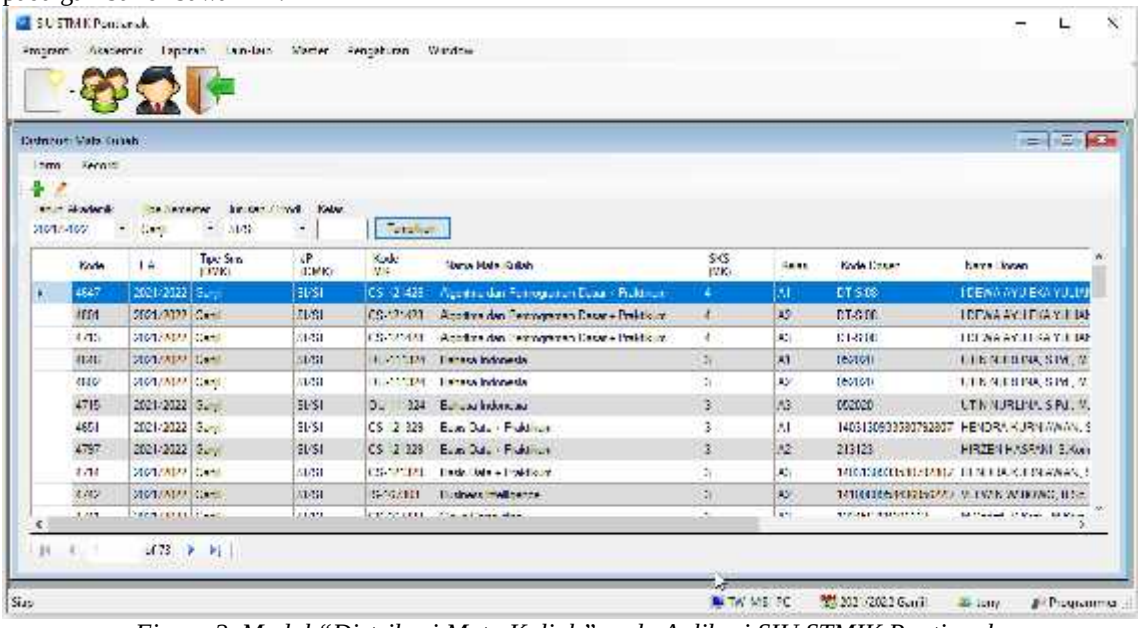

Figure 2. Modul "Distribusi Mata Kuliah" pada Aplikasi SIU STMIK Pontianak.

Gambar di atas menunjukkan modul Distribusi Mata Kuliah. Aplikasi desktop ini tidak langsung melakukan koneksi ke basis data di cloud, namun melewati sebuah web API dengan alamat https://stmikpontianak.net/ SiakStmikPtkApi/course_distribution/get. Tampilan coding untuk mengakses web API dapat dilihat pada gambar berikut :

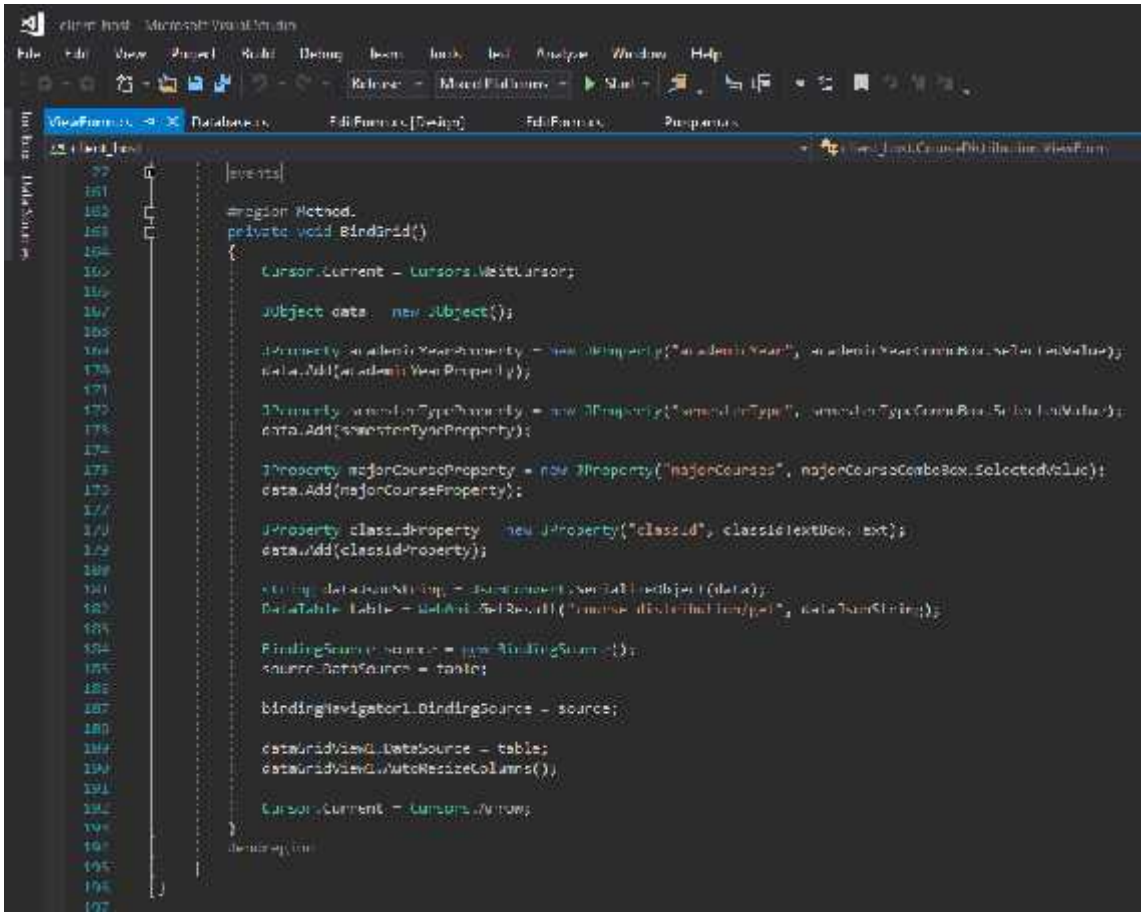

Figure 3. Coding untuk Mengakses Web API dalam Bahasa Pemrograman C#.

Tampilan pada gambar di atas merupakan tampilan IDE Visual Studio 2017 yang fokus pada fungsi BindGrid. Jobject merupakan objek dari library Newtonsoft yang berfungsi untuk menserialisasi objek-objek dari .NET Framework menjadi JSON, atau sebaliknya. Jpropery merupakan property yang akan dikirimkan melalui JSON ke server. Tipe data Jproperty hanya satu, yaitu object. Pada fungsi tersebut terdapat 4 variabel yang akan dikirim dalam format JSON yaitu :

- academicYear merupakan variabel yang berisi data tahun akademik,
- semesterType merupakan variabel yang berisi data tipe semester (ganjil / genap / SP),
- majorCourses merupakan variabel yang berisi data jurusan / program studi dan
- classId merupakan variabel yang berisi data kelas.

Setelah keempat Jproperty tersebut di atas dimasukkan ke Jobject, maka Jobject perlu diserialisasi menjadi tipe data string dengan bantuan library Newtonsoft dan disimpan dalam variabel dataJsonString. Fungsi yang dipanggil adalah JsonConvert.SerializeObject.

Data string tersebut kemudian dikirimkan ke web API dengan end point https://stmikpontianak.net/SiakStmikPtkApi/course_distribution/get. Web API akan membalas dengan memberikan data dalam format JSON. Data JSON tersebut diubah menjadi tipe data DataTable dengan bantuan library Newtonsoft pula. Fungsi tersebut adalah WebApi.GetResult. Dari DataTable tersebut kemudian diberikan kepada Binding Source supaya dapat didistribusikan kepada 2 komponen secara sinkronus yaitu : DataGridView dan Binding Navigator. Setelah data tersebut didistribusikan, maka layar akan menampilkan data dari web API ke layar monitor. Tampilannya dapat dilihat pada gambar 3 di atas.

Fungsi GetResult dari class WebApi dapat dilihat pada 2 (dua) gambar berikut :

Figure 4. Fungsi GetResult dari Class WebApi (Bagian 1).

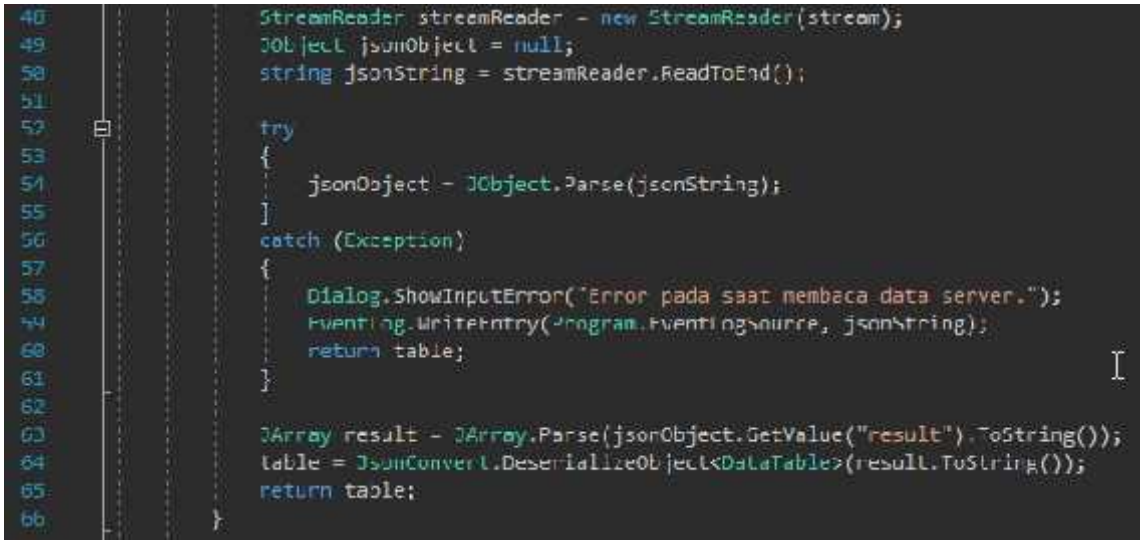

Figure 5. Fungsi GetResult dari Class WebApi (Bagian 2).

Variabel fullUrl merupakan gabungan antara variabel global Program.WebApiBaseUrl ditambah dengan variabel apiPath. Program.WebApiBaseUrl merupakan variabel global dari aplikasi yang bernilai "https://stmikpontianak.net/SiakStmikPtkApi/". Variabel api Path diberikan ketika mengakses fungsi GetResult. Pada gambar 4 dapat dilihat bahwa ketika mengakses fungsi Get Result, coding tersebut memberikan parameter bernilai "course_distribution/get". Maka apiPath pada fungsi Get Result memiliki nilai tersebut. Setelah digabungkan, maka variabel fullUrl bernilai "https://stmikpontianak.net/SiakStmikPtkApi/course_distribution/get". URL lengkap ini akan dikirimkan dengan metode POST dengan format JSON. Ketika dikirimkan, fungsi ini juga akan mengirimkan parameter yang telah diolah pada gambar 4 (variabel data Json String) dalam format JSON pula.

Web API akan membalas dengan memberikan data dalam format JSON. Berikut ditampilkan coding untuk web API untuk endpoint https://stmikpontianak.net/SiakStmikPtkApi/course_distribution/get :

application > controllers > course_distribution > Get.php > PHP Intelephense > Get > index_post

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

require APPPATH . '/libraries/REST_Controller.php';
require APPPATH . '/libraries/Format.php';

use Restserver\Libraries\REST_Controller;

class Get extends REST_Controller
{
    function index_post()
    {
        // Begin: Get data
        //$phpInput = file_get_contents('php://input');
        //$response['rawParameter'] = $phpInput;

        $academicYear = $this->post('academicYear');
        $academicYear = database_normalizeString($academicYear);

        $semesterType = $this->post('semesterType');
        $semesterType   database_normalizeString($semesterType);

        $majorCourses = $this->post('majorCourses');
        $majorCourses = database_normalizeString($majorCourses);

        $classId   $this->post("classId");
        $classId = database_normalizeString($classId);
        $classId = database_wildcardPad($classId);

        $courseDistributionId   (float) $this->post('courseDistributionId');

        $sql = "select ".
                "mmk_id As Kode, ".
                "tahun_akademik AS 'T.A.', TipeSemester AS 'Tipe Sms (DMK)', ".
                "mmk_jurusan_prodi AS 'JP (DMK)', ".
                "kode_mata_kuliah AS 'Kode MK', mmk_name AS 'Nama Mata Kuliah', mmk_sks AS 'SKS (MK)', ".
                "kelas As kelas, ".
                "kode_dosen AS 'Kode Dosen', mdl.md_nama AS 'Nama Dosen', ".
                "TotalKehadiran AS 'Total Hadir', UtsPadaPertemuan AS 'UTS pada', UasPadaPertemuan AS 'UAS pada', ".
                "IsClose ".
            "from distribusi_mata_kuliah ".
                "join master_mata_kuliah on distribusi_mata_kuliah.kode_mata_kuliah = master_mata_kuliah.mmk_id ".
                "INNER JOIN master_dosen mdl ON distribusi_mata_kuliah.kode_dosen = mdl.md_id ".
                "INNER JOIN master_kelas ON distribusi_mata_kuliah.kelas   master_kelas.mkls_id ".
            "WHERE ".
                "mkls_aktif = 1 AND ".
                "mkls_aktif = 'true' ";
```

Figure 6. Web API (Bagian 1)

Figure 7. Web API (Bagian 2)

Gambar di atas menunjukkan cara web API menerima input, mengakses basis data dan kemudian memformat-nya menjadi data JSON. Parameter yang diterima ada 5 yaitu :

- academicYear tahun akademik (Contoh : 2021/2022)
- semesterType tipe semester (Ganjil, Genap atau SP)
- majorCourses jurusan / prodi (Contoh : SI/SI)
- classId Kelas (Contoh : A1)
- courseDistributionId (Kode dari tabel 'distribusi_mata_kuliah')

Kelima variabel di atas sifatnya tidak wajib, jadi client tidak harus mengirimkan kelima-limanya secara lengkap, namun disesuaikan dengan kebutuhan saja. Setelah web API menerima input tersebut, selanjutnya akan dibuat syntax SQL yang dibutuhkan, berdasarkan input yang diterima. Kemudian syntax SQL tersebut akan dieksekusi oleh CodeIgniter ke basis data MariaDB dan menerima hasilnya. CodeIgniter kemudian akan memformat data dari basis data menjadi JSON.

Data JSON kemudian akan dikembalikan ke client untuk diproses menjadi tampilan bagi pemakai. Cara membacanya adalah dengan menggunakan objek StreamReader dari library .NET Framework. Data hasil baca dari StreamReader masih berupa string. Kita perlu menjadi Jarray dengan bantuan library Newtonsoft. Setelah menjadi Jarray, maka data tersebut dapat diserialisasi menjadi tipe data DataTable. DataTable inilah yang akan dikembalikan kepada pemanggilnya supaya dapat diolah lebih lanjut.

Pengujian dilakuan dengan memasukkan 2 baris coding untuk mencatat :

- Tanggal dan waktu dimulainya fungsi BindGrid()
- Tanggal dan waktu selesainya fungsi BindGrid()

Data tersebut di atas dituliskan pada Event Viewer yang merupakan aplikasi bawaan pada setiap sistem operasi Windows. Dari data tersebut kita akan dapat membandingkan kecepatan muat data antara sistem web API dengan sistem connection pooling. Presisi yang digunakan adalah hingga milidetik. Tujuannya adalah supaya mendapat tingkat presisi yang sangat tinggi. Format yang digunakan dalam bahasa C# adalah “dd-MM-yyyy HH:mm:ss:ffff”. Data yang ditampilkan ada sebanyak 37 record dari web API. Kedua baris coding yang ditambahkan dapat dilihat pada gambar berikut :

```
private void DisGrid()
{
    EventLog.WriteEntry("Program EventLogSource", "Aksi dimulai pada : " + DateTime.Now.ToString("dd-MM-yyyy HH:mm:ss:ffff"));
    Cursor.Current = Cursors.WaitCursor;

    JObject data = new JObject();

    JProperty academicYearProperty = new JProperty("academicYear", academicYearComboBox.SelectedValue);
    data.Add(academicYearProperty);

    JProperty semesterTypeProperty = new JProperty("semesterType", semesterTypeComboBox.SelectedValue);
    data.Add(semesterTypeProperty);

    JProperty majorCourseProperty = new JProperty("majorCourses", majorCourseComboBox.SelectedValue);
    data.Add(majorCourseProperty);

    JProperty classIdProperty = new JProperty("classId", classIdTextBox.Text);
    data.Add(classIdProperty);

    string dataJsonString = JsonConvert.SerializeObject(data);
    DataTable table = WebAPI.GetResult("course_distribution/get", dataJsonString);

    BindingSource source = new BindingSource();
    source.DataSource = table;

    bindingNavigator1.BindingSource = source;

    dataGridView1.DataSource = table;
    dataGridView1.AutoResizeColumns();

    Cursor.Current = Cursors.Arrow;
    EventLog.WriteEntry("Program EventLogSource", "Aksi selesai pada : " + DateTime.Now.ToString("dd-MM-yyyy HH:mm:ss:ffff"));
}
```

Figure 8. Dua Baris Coding Tambahan untuk Pengujian.

Hasil dari coding di atas dapat dilhat pada Event Viewer seperti pada kedua gambar berikut :

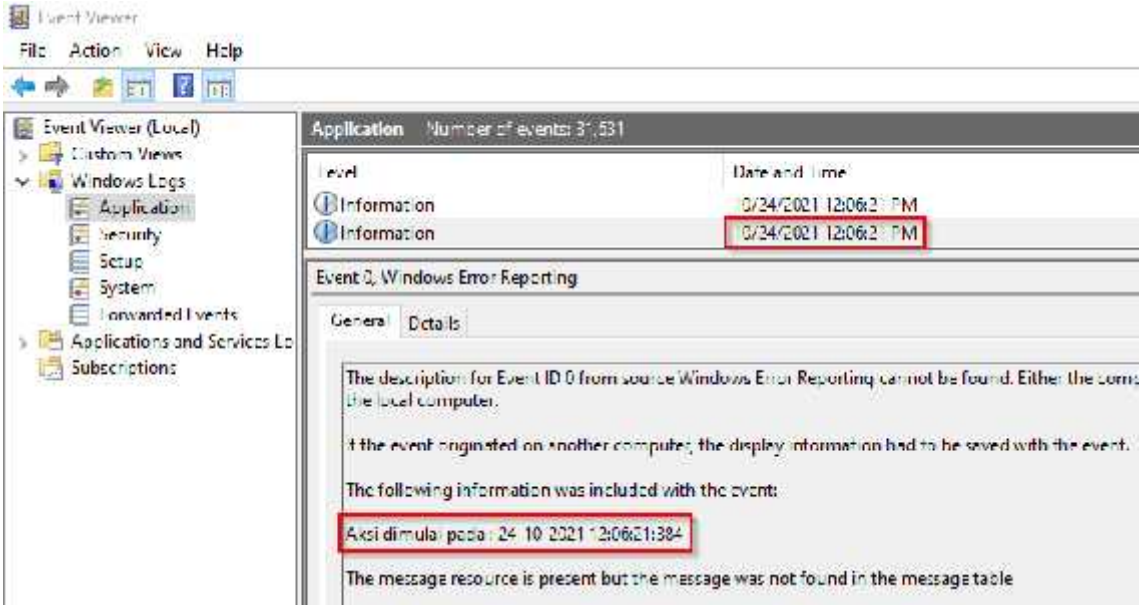

Figure 9. Event Viewer yang Menunjukkan Aksi Dimulai.

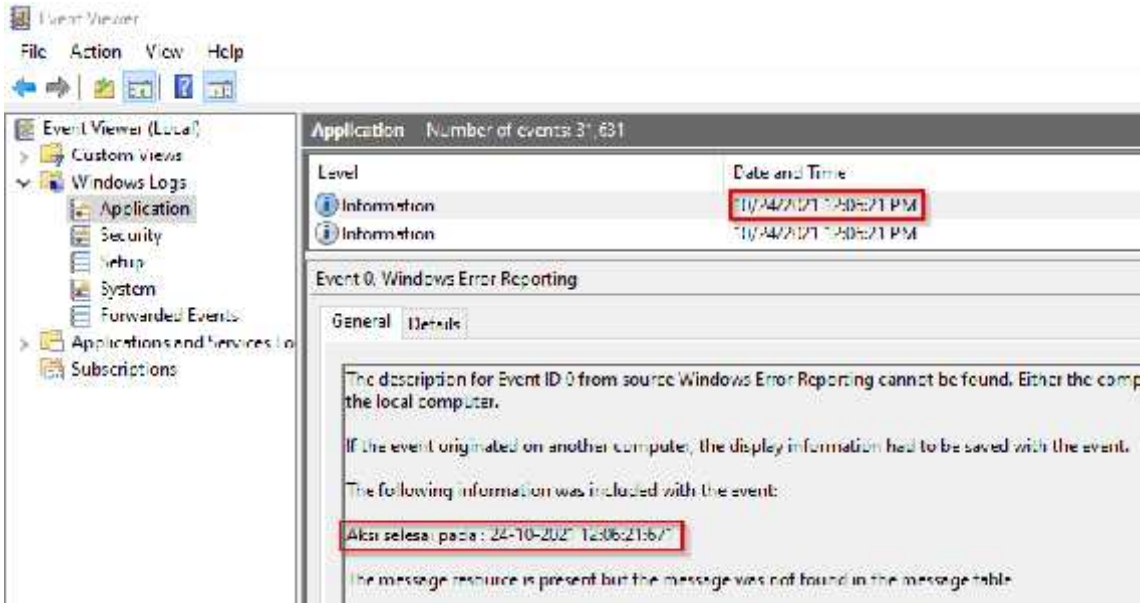

Figure 10. Event Viewer yang Menunjukkan Aksi Selesai.

Dari kedua gambar tersebut di atas, dapat dilihat bahwa aksi dimulai pada 24 Oktober 2021 pukul 12.06 WIB pada detik ke-21, milidetik ke-384. Sedangkan aksi tersebut selesai, mulai dari mengakses web API, menerima kembali data dalam format JSON, mengkonversi menjadi DataTable dan menampilkan ke DataGridView, selesai pada 24 Oktober 2021 pukul 12.06 WIB pada detik ke-21, milidetik ke-671. Ini merupakan performa yang sangat luar biasa, karena semua ini terjadi dalam kurang dari 1 detik. Secara presisi, proses ini terjadi hanya dalam waktu 287 milidetik atau kurang lebih 0.3 detik saja. Sedang untuk teknik connection pooling pada penelitian sebelumnya, untuk 73 record dibutuhkan 5.05 detik. Jadi apabila diambil perbandingan antara web API dengan teknik connection pooling, maka kecepatan eksekusi adalah 1 : 16. Ini membuktikan bahwa performa dari web API 16 kali lipat lebih laju dibandingkan dengan connection pooling. Hal ini merupakan peningkatan performa yang sangat signifikan.

4. Kesimpulan

Berdasarkan hasil dan pembahasan, maka dapat diambil kesimpulan bahwa implementasi web API pada aplikasi desktop SIU STMIK Pontianak berhasil meningkatkan performa aplikasi secara signifikan apabila dibandingkan dengan teknik connection pooling. Hal ini berarti fitur ini dapat membantu meningkatkan produktifitas para staf akademik pada saat melaksanakan tugasnya.

Daftar Pustaka

[1] T. Wijaya, "Teknik Connection Pooling untuk Meningkatan Performa Aplikasi dengan Basis Data Jarak Jauh," Seminar Nasional Sistem Informasi dan Teknik Informatik (SENSITIF), vol. 1, no. 1, pp. 153-160, 2019.
[2] Microsoft Docs, "SQL Server Connection Pooling," Microsoft, 30 3 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql-server-connection-pooling. [Accessed 15 9 2019].
[3] H. Sacham and D. Boneh, "Improving SSL Handshake Performance via Batching," Cryptographers' Track at The RSA Conference, pp. 28-43, April 2001.
[4] V. Singh, U. V. Sawanat, P. GOEL and N. G. Deshaveni, "Method and system for transparent database connection pooling and query queuing". United States Patent US8484242B1, 9 July 2013.
[5] C. Dewi, K. H. B. Putro and R. Somya, "Implementasi Sistem Klaim Asuransi Kendaraan Bermotor di PT. IBS Jakarta Berbasis Mobile," CCIT Journal, vol. 9, no. 2, pp. 191-202, 14 Januari 2016.
[6] W. Purhantara, Metode Penelitian Kualitatif untuk Bisnis, Yogyakarta: Graha Ilmu, 2010.
[7] A. A. Albarqi, "The Proposed L-Scrumban Methodology to Improve the Efficiency of Agile Software Development," I.J. Information Engineering and Electronic Business, vol. 3, p. 13, 2018.
[8] M. Beedle, A. v. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, K. Schwaber, J. Sutherland and D. Thomas, "Signatories:

The Agile Manifesto," 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed 8 September 2018].
[9] L. Lindstrom and R. Jeffries, "Extreme Programming and Agile Software Development Methodologies," Information Systems Management, pp. 41-52, 2004.
[10] Microsoft Docs, "Overview of the .NET Framework," Microsoft, 30 March 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview. [Accessed 8 September 2018].
[11] Andy (Steve) De George, "What is Windows Forms," Microsoft, 26 10 2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-5.0. [Accessed 28 09 2021].
[12] Bill Wagner, "A Tour of C#," Microsoft, 23 08 2021. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Accessed 28 09 2021].
[13] Microsoft Developer Network (MSDN), "Visual Studio IDE," Microsoft, 23 August 2018. [Online]. Available: https://msdn.microsoft.com/en-us/library/dn762121.aspx. [Accessed 8 September 2018].
[14] "The leading operating system for PCs, IoT devices, servers and the cloud," Ubuntu, [Online]. Available: https://ubuntu.com. [Accessed 15 09 2019].
[15] "Supporting continuity and open collaboration," MariaDB, 2019. [Online]. Available: https://mariadb.org. [Accessed 15 09 2019]..