ДОДАТОК А

Слайди презентації

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

АТЕСТАЦІЙНА РОБОТА БАКАЛАВРА

Дослідження методів обробки даних для отримання температурних показників комплектуючих комп'ютера

Керівник проекту:
проф. Власенко Л.А.

Виконав
студент групи ІПЗм-18-2
Білобров Є.О.

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА АКТУАЛІЗАЦІЯ РІШЕНЬ

Бездоганна робота персонального комп'ютера не може не радувати свого власника, але не завжди ЕОМ поводиться належним чином. Симптоми і причини поганої роботи можуть бути різними. Перегрів процесора і інших компонентів ПК - одна з найбільш частих проблем виходу з ладу комп'ютера. До того ж, знову прийшло літо, а це значить, що температура повітря відчутно підскочила. І це відчувають не тільки люди, але і наші з Вами комп'ютери, яким і без того доводиться жарко, а тут ще й за вікном сонечко припікає. Коли нам жарко, що з нами відбувається? Правильно, в кращому випадку нам просто погано і дискомфортно, перестає нормально думатиметься, а в гіршому - ми ловимо сонячний удар.

Метою роботи дослідження методів обробки даних для отримання температурних показників комплектуючих комп'ютера та впровадження програми контролю температури з використанням технології IONIC4, C#, бази даних MongoDB, локально – розгорнутого серверу Nest.js; відпрацювання основних принципів написання коду, основ рефакторінгу коду, а також загальних принципів розробки, проектування та підтримки програмного забезпечення; закріплення теоретичних знань та практичних навичок, що були набуті в ході вивчення ряду навчальних дисциплін.

В рамках роботи над серверною частиною для системи мають бути підтверджені навики що стосуються проектування, розробки. Також навики, що стосуються аналізу обраної предметної області та навики написання програмного забезпечення в цілому.

Постановка задачі

Метою роботи дослідження методів обробки даних для отримання температурних показників комплектуючих комп'ютера та впровадження програми контролю температури з використанням технології IONIC4, C#, бази даних MongoDB, локально – розгорнутого серверу Nest.js; відпрацювання основних принципів написання коду, основ рефакторінгу коду, а також загальних принципів розробки, проектування та підтримки програмного забезпечення; закріплення теоретичних знань та практичних навичок, що були набуті в ході вивчення ряду навчальних дисциплін.

В рамках роботи над серверною частиною для системи мають бути підтверджені навики що стосуються проектування, розробки. Також навики, що стосуються аналізу обраної предметної області та навики написання програмного забезпечення в цілому.

Існуючи аналоги

1.Утиліта Game Assistant:

+ Зручний інтерфейс
+ Відображення температури з боку екрана
- Додавання ігор в ручному режимі
- Додаток актуален тільки для геймерів

2.Утиліта AIDA64

+ Відображається вся інформація про датчики
+ Присутня інформація про напругу комплектуючих
- Має досить високу ціну
- Недостатньо інтуїтивне управління програмою.

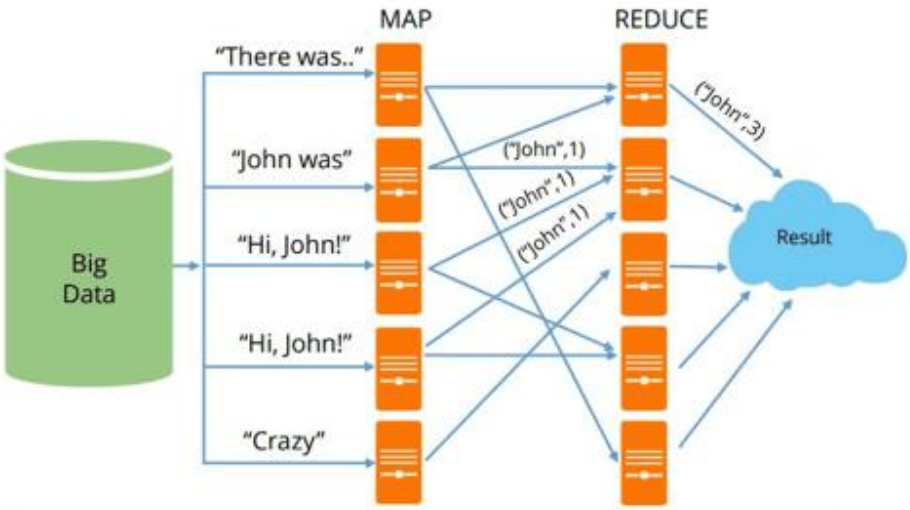
Технології

- C#
- Nest.js
- JSON
- MongoDB
- Angular та IONIC4

Перш за все треба розглянути що таке Big Data - це серія підходів, інструментів і методів обробки структурованих, слабо структурованих і неструктурованих даних величезних обсягів і значного різноманіття для отримання сприймаються людиною результатів, ефективних в умовах безперервного приросту, розподілу по численних вузлів обчислювальної мережі, альтернативних традиційним системам управління базами даних і рішень класу Business Intelligence . В дану серію включають кошти масово-паралельної обробки невизначено структурованих даних, перш за все, рішеннями категорії NoSQL, алгоритмами MapReduce, програмними каркасами і бібліотеками проекту Hadoop

Что же такое **BIG DATA?**

- Big Data — это наборы данных такого объема, что традиционные инструменты не способны осуществлять их захват, управление и обработку за приемлемое для практики время.
- Технология Big Data предоставляет услуги, помогающие раскрыть коммерческий потенциал мегамассивов данных за счет поиска ценных закономерностей и фактов путем объединения и анализа больших объемов данных.
- В качестве определяющих характеристик для больших данных выделяют «три V»:

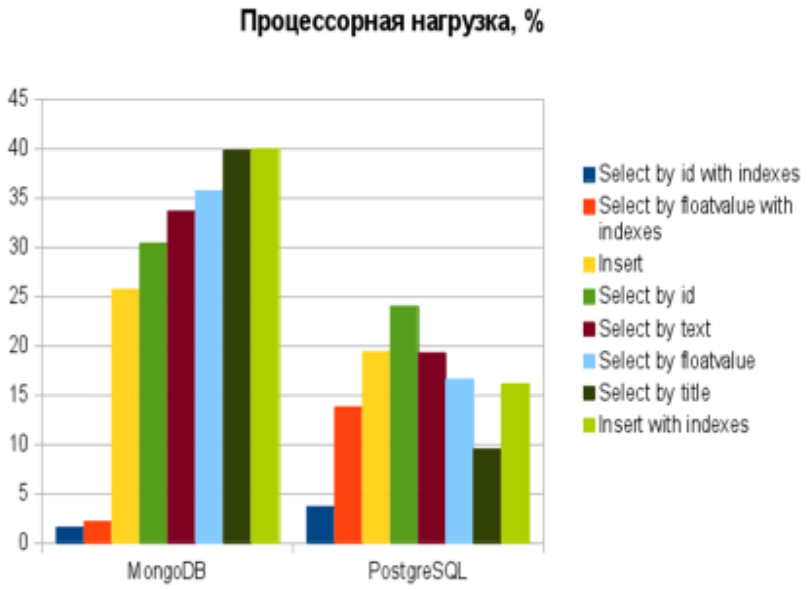


Что есть в MongoDB:

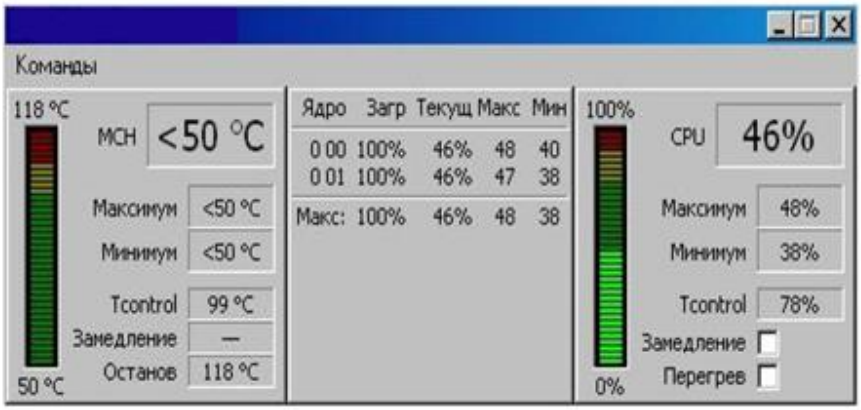
- гибкий язык для формирования запросов
- динамические запросы
- индексация коллекций
- профилирование запросов
- журналирование операций записи
- отказоустойчивость и масштабируемость
- асинхронная репликация и шардинг
- MapReduce
- полнотекстовый поиск с поддержкой морфологии

Почему не SQL?

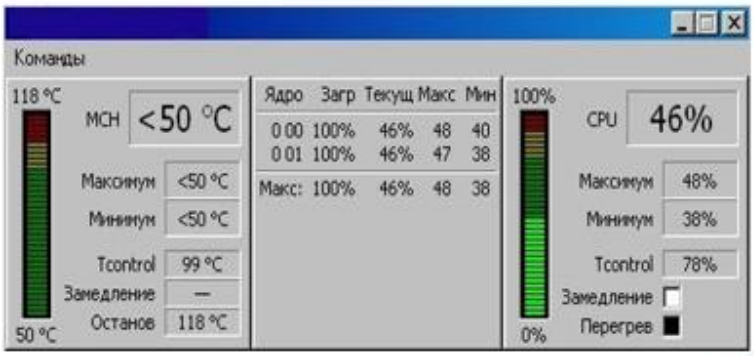
- Сложность создания запросов на SQL, которые выходят за рамки простого SELECT.
- Посредственная масштабируемость на сверхвысокую нагрузку в виде большого количества запросов на чтение и запись
- Плохая адаптируемость к сетям распределённых вычислений.

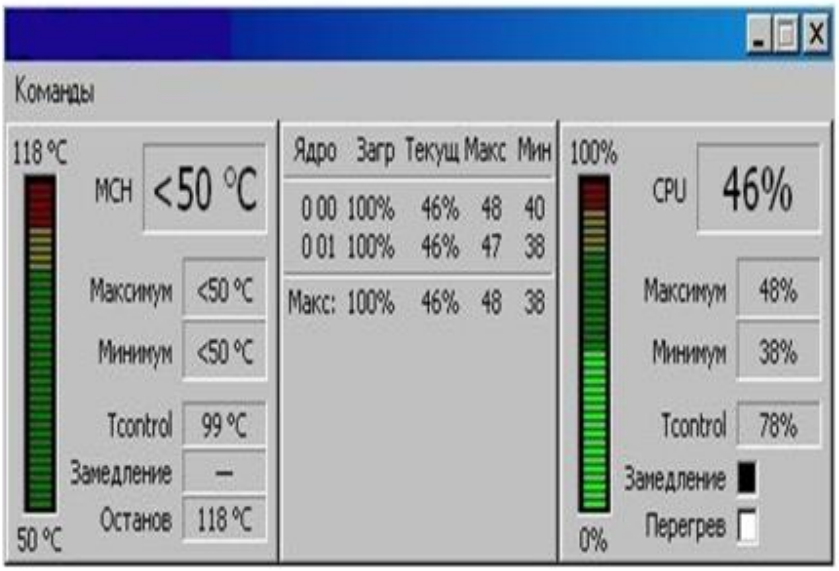


ІНТЕРФЕЙС ТА ФУНКЦІОНАЛ



За допомогою цієї утиліти можливо не тільки моніторити показники,,а саме завантаженість, поточну, максимальну чи мінімальну завантаженість у процентах. Також можливо убезпечити свій комп'ютер від перегріву зробивши позначку з боку від "Перегрев" як на рисунку





ВИСНОВКИ

В результаті виконання атестаційної роботи було спроектовано та розроблено серверну частину для системи контролю температур, що взаємодіє із зчитувачами та BIOS'ом . Відзначимо, що серверна частина цілком відповідає вимогам та набору функцій, що висувалися до неї, обробляє інформацію, отриману із зчитувача та віддає її користувачеві в режимі реального часу.

Була досліджена і реалізована схема бази даних в PostgreSQL і MongoDB, що дозволяє реалізувати складні запити і порівняти вищезгадані СУБД на рівні завдань, в якості яких виступили аналітичні запити під час перегляду цього всю базу даних. Були отримані результати вимірювань часу виконання запитів на "Гарячої" базі даних, які показують, що для даного класу задач MongoDB є більш ефективною, ніж PostgreSQL.

ДОДАТОК Б

Лістинг коду

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Security.Permissions;
using System.Reflection;

namespace OpenHardwareMonitor.Hardware {

  public class Computer : IComputer {

    private readonly List<IGroup> groups = new List<IGroup>();
    private readonly ISettings settings;

    private SMBIOS smbios;

    private bool open;

    private bool mainboardEnabled;
    private bool cpuEnabled;
    private bool ramEnabled;
    private bool gpuEnabled;
    private bool fanControllerEnabled;
    private bool hddEnabled;

    public Computer() {
      this.settings = new Settings();
    }

    public Computer(ISettings settings) {
      this.settings = settings ?? new Settings();
    }

    private void Add(IGroup group) {
      if (groups.Contains(group))
        return;

      groups.Add(group);

      if (HardwareAdded != null)
        foreach (IHardware hardware in group.Hardware)
          HardwareAdded(hardware);
    }

    private void Remove(IGroup group) {
      if (!groups.Contains(group))
        return;
```

```
    groups.Remove(group);

    if (HardwareRemoved != null)
      foreach (IHardware hardware in group.Hardware)
        HardwareRemoved(hardware);

    group.Close();
  }

  private void RemoveType<T>() where T : IGroup {
    List<IGroup> list = new List<IGroup>();
    foreach (IGroup group in groups)
      if (group is T)
        list.Add(group);
    foreach (IGroup group in list)
      Remove(group);
  }

  [SecurityPermission(SecurityAction.LinkDemand, UnmanagedCode = true)]
  public void Open() {
    if (open)
      return;

    this.smbios = new SMBIOS();

    Ring0.Open();
    Opcode.Open();

    if (mainboardEnabled)
      Add(new Mainboard.MainboardGroup(smbios, settings));

    if (cpuEnabled)
      Add(new CPU.CPUGroup(settings));

    if (ramEnabled)
      Add(new RAM.RAMGroup(smbios, settings));

    if (gpuEnabled) {
      Add(new ATI.ATIGroup(settings));
      Add(new Nvidia.NvidiaGroup(settings));
    }

    if (fanControllerEnabled) {
      Add(new TBalancer.TBalancerGroup(settings));
      Add(new Heatmaster.HeatmasterGroup(settings));
    }

    if (hddEnabled)
      Add(new HDD.HarddriveGroup(settings));

    open = true;
  }
```

```
    public bool MainboardEnabled {
      get { return mainboardEnabled; }

      [SecurityPermission(SecurityAction.LinkDemand, UnmanagedCode = true)]
      set {
        if (open && value != mainboardEnabled) {
          if (value)
            Add(new Mainboard.MainboardGroup(smbios, settings));
          else
            RemoveType<Mainboard.MainboardGroup>();
        }
        mainboardEnabled = value;
      }
    }

    public bool CPUEnabled {
      get { return cpuEnabled; }

      [SecurityPermission(SecurityAction.LinkDemand, UnmanagedCode = true)]
      set {
        if (open && value != cpuEnabled) {
          if (value)
            Add(new CPU.CPUGroup(settings));
          else
            RemoveType<CPU.CPUGroup>();
        }
        cpuEnabled = value;
      }
    }

    public bool RAMEnabled {
      get { return ramEnabled; }

      [SecurityPermission(SecurityAction.LinkDemand, UnmanagedCode = true)]
      set {
        if (open && value != ramEnabled) {
          if (value)
            Add(new RAM.RAMGroup(smbios, settings));
          else
            RemoveType<RAM.RAMGroup>();
        }
        ramEnabled = value;
      }
    }

    public bool GPUEnabled {
      get { return gpuEnabled; }

      [SecurityPermission(SecurityAction.LinkDemand, UnmanagedCode = true)]
      set {
        if (open && value != gpuEnabled) {
          if (value) {
            Add(new ATI.ATIGroup(settings));
            Add(new Nvidia.NvidiaGroup(settings));
```

```
        } else {
          RemoveType<ATI.ATIGroup>();
          RemoveType<Nvidia.NvidiaGroup>();
        }
      }
      gpuEnabled = value;
    }
  }

  public bool FanControllerEnabled {
    get { return fanControllerEnabled; }

    [SecurityPermission(SecurityAction.LinkDemand, UnmanagedCode = true)]
    set {
      if (open && value != fanControllerEnabled) {
        if (value) {
          Add(new TBalancer.TBalancerGroup(settings));
          Add(new Heatmaster.HeatmasterGroup(settings));
        } else {
          RemoveType<TBalancer.TBalancerGroup>();
          RemoveType<Heatmaster.HeatmasterGroup>();
        }
      }
      fanControllerEnabled = value;
    }
  }

  public bool HDDEnabled {
    get { return hddEnabled; }

    [SecurityPermission(SecurityAction.LinkDemand, UnmanagedCode = true)]
    set {
      if (open && value != hddEnabled) {
        if (value)
          Add(new HDD.HarddriveGroup(settings));
        else
          RemoveType<HDD.HarddriveGroup>();
      }
      hddEnabled = value;
    }
  }

  public IHardware[] Hardware {
    get {
      List<IHardware> list = new List<IHardware>();
      foreach (IGroup group in groups)
        foreach (IHardware hardware in group.Hardware)
          list.Add(hardware);
      return list.ToArray();
    }
  }

  private static void NewSection(TextWriter writer) {
    for (int i = 0; i < 8; i++)
```

```
      writer.Write("----------");
    writer.WriteLine();
    writer.WriteLine();
  }

  private static int CompareSensor(ISensor a, ISensor b) {
    int c = a.SensorType.CompareTo(b.SensorType);
    if (c == 0)
      return a.Index.CompareTo(b.Index);
    else
      return c;
  }

  private static void ReportHardwareSensorTree(
    IHardware hardware, TextWriter w, string space)
  {
    w.WriteLine("{0}|", space);
    w.WriteLine("{0}+- {1} ({2})",
      space, hardware.Name, hardware.Identifier);
    ISensor[] sensors = hardware.Sensors;
    Array.Sort(sensors, CompareSensor);
    foreach (ISensor sensor in sensors) {
      w.WriteLine("{0}|  +- {1,-14} : {2,8:G6} {3,8:G6} {4,8:G6} ({5})",
        space, sensor.Name, sensor.Value, sensor.Min, sensor.Max,
        sensor.Identifier);
    }
    foreach (IHardware subHardware in hardware.SubHardware)
      ReportHardwareSensorTree(subHardware, w, "|  ");
  }

  private static void ReportHardwareParameterTree(
    IHardware hardware, TextWriter w, string space) {
    w.WriteLine("{0}|", space);
    w.WriteLine("{0}+- {1} ({2})",
      space, hardware.Name, hardware.Identifier);
    ISensor[] sensors = hardware.Sensors;
    Array.Sort(sensors, CompareSensor);
    foreach (ISensor sensor in sensors) {
      string innerSpace = space + "|  ";
      if (sensor.Parameters.Length > 0) {
        w.WriteLine("{0}|", innerSpace);
        w.WriteLine("{0}+- {1} ({2})",
          innerSpace, sensor.Name, sensor.Identifier);
        foreach (IParameter parameter in sensor.Parameters) {
          string innerInnerSpace = innerSpace + "|  ";
          w.WriteLine("{0}+- {1} : {2}",
            innerInnerSpace, parameter.Name,
            string.Format(CultureInfo.InvariantCulture, "{0} : {1}",
              parameter.DefaultValue, parameter.Value));
        }
      }
    }
    foreach (IHardware subHardware in hardware.SubHardware)
      ReportHardwareParameterTree(subHardware, w, "|  ");
```

```
  }

  private static void ReportHardware(IHardware hardware, TextWriter w) {
    string hardwareReport = hardware.GetReport();
    if (!string.IsNullOrEmpty(hardwareReport)) {
      NewSection(w);
      w.Write(hardwareReport);
    }
    foreach (IHardware subHardware in hardware.SubHardware)
      ReportHardware(subHardware, w);
  }

  public string GetReport() {

    using (StringWriter w = new StringWriter(CultureInfo.InvariantCulture)) {

      w.WriteLine();
      w.WriteLine("Open Hardware Monitor Report");
      w.WriteLine();

      Version version = typeof(Computer).Assembly.GetName().Version;

      NewSection(w);
      w.Write("Version: "); w.WriteLine(version.ToString());
      w.WriteLine();

      NewSection(w);
      w.Write("Common Language Runtime: ");
      w.WriteLine(Environment.Version.ToString());
      w.Write("Operating System: ");
      w.WriteLine(Environment.OSVersion.ToString());
      w.Write("Process Type: ");
      w.WriteLine(IntPtr.Size == 4 ? "32-Bit" : "64-Bit");
      w.WriteLine();

      string r = Ring0.GetReport();
      if (r != null) {
        NewSection(w);
        w.Write(r);
        w.WriteLine();
      }

      NewSection(w);
      w.WriteLine("Sensors");
      w.WriteLine();
      foreach (IGroup group in groups) {
        foreach (IHardware hardware in group.Hardware)
          ReportHardwareSensorTree(hardware, w, "");
      }
      w.WriteLine();

      NewSection(w);
      w.WriteLine("Parameters");
      w.WriteLine();
```

```
      foreach (IGroup group in groups) {
        foreach (IHardware hardware in group.Hardware)
          ReportHardwareParameterTree(hardware, w, "");
      }
      w.WriteLine();

      foreach (IGroup group in groups) {
        string report = group.GetReport();
        if (!string.IsNullOrEmpty(report)) {
          NewSection(w);
          w.Write(report);
        }

        IHardware[] hardwareArray = group.Hardware;
        foreach (IHardware hardware in hardwareArray)
          ReportHardware(hardware, w);

      }
      return w.ToString();
    }
  }

  public void Close() {
    if (!open)
      return;

    while (groups.Count > 0) {
      IGroup group = groups[groups.Count - 1];
      Remove(group);
    }

    Opcode.Close();
    Ring0.Close();

    this.smbios = null;

    open = false;
  }

  public event HardwareEventHandler HardwareAdded;
  public event HardwareEventHandler HardwareRemoved;

  public void Accept(IVisitor visitor) {
    if (visitor == null)
      throw new ArgumentNullException("visitor");
    visitor.VisitComputer(this);
  }

  public void Traverse(IVisitor visitor) {
    foreach (IGroup group in groups)
      foreach (IHardware hardware in group.Hardware)
        hardware.Accept(visitor);
  }
```

```
    private class Settings : ISettings {

      public bool Contains(string name) {
        return false;
      }

      public void SetValue(string name, string value) { }

      public string GetValue(string name, string value) {
        return value;
      }

      public void Remove(string name) { }
    }
  }
}
```