Architectural Optimization of Windows 11 for High-Performance Edge AI Workloads on Constrained Hardware

1. Executive Analysis of Computational Constraints and Strategic Objectives

The democratization of Artificial Intelligence (AI) and Machine Learning (ML) development has historically presumed the availability of data-center-grade infrastructure or high-end consumer workstations equipped with expansive Video Random Access Memory (VRAM) and system memory. However, a significant cohort of practitioners operates on mobile workstations—specifically, enterprise-grade laptops like the Lenovo ThinkPad X1 Extreme Gen 4i—which, despite their premium pedigree, face severe resource contention when tasked with modern generative AI workloads. This report addresses the specific architectural challenge of maximizing available Random Access Memory (RAM) and optimizing the NVIDIA RTX 3050 Ti GPU (4GB VRAM) within a Windows 11 Pro environment to support a sophisticated stack comprising Whisper (Speech-to-Text), Kokoro (Text-to-Speech), embedding models, and vector database operations.

The target system presents a strict hardware envelope: 16GB of non-upgradable DDR4 RAM and an RTX 3050 Ti. In a default Windows 11 configuration, the Operating System (OS) overhead, combined with background telemetry, manufacturer-specific bloatware, and inefficient virtualization configurations, typically leaves a meager 2-3GB of RAM available for user applications. This is insufficient for loading even quantized Large Language Models (LLMs) or managing the memory pressure of the Windows Subsystem for Linux (WSL2) alongside a modern Integrated Development Environment (IDE). The objective is to reclaim sufficient system resources to achieve a stable baseline of 4-6GB of free RAM, creating a viable environment for edge AI development.

Achieving this requires a systemic overhaul moving beyond superficial "tuning." It necessitates a deep intervention into the Windows Service Control Manager, the manipulation of the Windows Registry to correct driver-level memory leaks, the precise configuration of the Hyper-V hypervisor managing WSL2, and the strategic quantization of AI models to fit within the 4GB VRAM hard limit. The following analysis provides a comprehensive, technical roadmap for transforming a constrained mobile workstation into an efficient edge AI node.

---

2. Windows 11 Service Architecture and Kernel-Level

Memory Reclamation

Windows 11 is engineered as a general-purpose operating system designed to prioritize feature readiness, rapid application launching, and seamless background synchronization over raw resource availability. For a dedicated AI development machine, this default posture represents a significant inefficiency. A granular audit of the Windows Service architecture reveals that gigabytes of committed memory are frequently consumed by processes irrelevant to the target workload.

2.1. The Non-Paged Pool and the NDU Memory Leak Mechanics

A critical, often overlooked component of Windows memory management is the Non-Paged Pool—a region of system memory guaranteed to reside in physical RAM at all times, never to be swapped to the page file. This pool is reserved for kernel-mode drivers and subsystems that cannot tolerate page faults. In Windows 8, 10, and 11, a primary contributor to the pathological inflation of this pool is the **Network Data Usage (NDU)** monitoring driver (Ndu.sys).

The NDU driver is responsible for tracking network traffic data to populate the "Data Usage" statistics in the Windows Settings panel. However, technical analysis indicates that this driver frequently exhibits behavior indistinguishable from a memory leak, particularly during high-bandwidth scenarios common in AI development, such as downloading multi-gigabyte model weights (e.g., .bin or .safetensors files) or pulling large Docker container images.[1] The driver allocates memory handles to track packet metadata but fails to release them effectively, leading to a silent, cumulative reduction in available RAM. This memory usage is often not attributed to a specific user-mode process in Task Manager, appearing instead as "System" memory or an inflated Non-Paged Pool in tools like RAMMap.[2]

Remediation Strategy:
Disabling the NDU service is a high-impact, low-risk optimization. It does not compromise network connectivity but merely disables the data usage tracking heuristic. This modification typically reclaims between 200MB and over 1GB of RAM depending on system uptime and cumulative network activity.2

2.2. Comprehensive Service Audit and Dependency Analysis

To maximize memory headroom, a rigorous audit of Windows services is required. The following analysis categorizes services based on their necessity for an AI/WSL2 workflow versus their contribution to system bloat, identifying dependencies that must be preserved to avoid system instability.

2.2.1. Telemetry, Diagnostics, and Update Services

These services are primarily responsible for data collection and background maintenance. On a constrained system, the CPU cycles and memory pages they consume are better allocated to the Python interpreter or the WSL2 VM.

Service Name	Display Name	Recommendation	Dependency/Risk Analysis
DiagTrack	Connected User Experiences and Telemetry	**Disable**	Safe. Primary telemetry collector. Disabling stops data reporting to Microsoft and frees 5-15MB RAM.[4]
DPS	Diagnostic Policy Service	**Disable**	Disabling prevents the generation of battery reports via powercfg /batteryreport. If daily battery diagnostics are not required, this is safe.[4]
WSearch	Windows Search	**Disable**	Stops file indexing. Improves I/O performance significantly. Search in Explorer will be slower and non-indexed, but essential for reclaiming ~100MB+ RAM.[5]
SysMain	SysMain (Superfetch)	**Disable**	Preloads apps into RAM. On 16GB systems with NVMe, the launch speed gain is negligible compared to the memory cost.

			Disabling frees Standby List pressure.[4]
MapsBroker	Downloaded Maps Manager	**Disable**	Unnecessary unless using built-in Maps applications. Safe to disable.
PcaSvc	Program Compatibility Assistant	**Disable**	Monitors for compatibility issues. Unnecessary overhead for a controlled dev environment.
WaasMedicSvc	Windows Update Medic Service	**Manual**	Attempts to repair Windows Update. Can be resource-heavy. Safe to set to Manual.[4]

2.2.2. Hardware and Peripheral Support Services

Disabling these requires careful consideration of the specific hardware in use (Bluetooth peripherals, biometrics).

Service Name	**Display Name**	**Recommendation**	**Dependency/Risk Analysis**
TabletInputService	Touch Keyboard and Handwriting Panel	**ENABLE**	**CRITICAL:** Required for Windows Terminal functionality on Windows 11. Disabling causes Terminal input failures.[4]

BTAGService	Bluetooth Audio Gateway Service	**Manual**	Required for Bluetooth audio/Hands-Free profile. Disable only if using wired audio exclusively.[4]
BthAvctpSvc	AVCTP Service	**Manual**	Required for Bluetooth audio control. Safe to disable if no Bluetooth audio devices are used.[4]
WbioSrvc	Windows Biometric Service	**Enable**	Required for Fingerprint reader on the ThinkPad X1. Disable only if not using biometric login.[4]
NlaSvc	Network Location Awareness	**Enable**	**CRITICAL:** Disabling breaks network status icons and can cause connectivity issues for dependent services.[4]

2.2.3. Third-Party and Manufacturer Bloatware

Lenovo devices often ship with services that duplicate OS functionality or provide unneeded "optimization."

- **Lenovo Vantage Service:** While useful for firmware updates, it runs background processes. It is recommended to set this to "Manual" and only start it explicitly when checking for BIOS updates.
- **McAfee / Third-Party Antivirus:** These are notoriously heavy on non-paged pool memory. Relying on Windows Defender is significantly more efficient. Removal tools (like MCPR for McAfee) should be used to ensure driver-level removal.[6]

2.3. The Standby List Problem and Automated Mitigation

Windows memory management adheres to a philosophy that "unused RAM is wasted RAM." Consequently, data that is read from the disk is cached in the **Standby List**. While this memory is technically marked as "available" (meaning the OS can discard it if a process needs RAM), the transition is not instantaneous. In high-pressure scenarios—such as loading a 4GB language model into RAM—the latency introduced by clearing the Standby List can cause stuttering or, in the context of WSL2, allocation failures where the Linux kernel perceives the host as out of memory.

Native Windows APIs allow for the programmatic clearing of this list without resorting to questionable third-party executables like EmptyStandbyList.exe, which has been flagged by some security vendors as Potentially Unwanted Programs (PUP).[8] The NtSetSystemInformation system call in ntdll.dll can be invoked via PowerShell to purge the standby list or working sets directly.[10]

Automated Implementation Strategy:
Rather than running a continuous background task (which itself consumes resources), a scheduled task that runs only when memory pressure is high or on a timer is preferable. The logic utilizes C# reflection within PowerShell to access the native API, bypassing the need for external binaries. This ensures a "living off the land" approach that is secure and efficient.11

---

3. The Windows Subsystem for Linux (WSL2) Hypervisor Architecture

The Windows Subsystem for Linux version 2 (WSL2) represents a paradigm shift from the translation layer of WSL1. It operates as a lightweight utility virtual machine (VM) running a real Linux kernel on top of the Hyper-V hypervisor. While this provides full system call compatibility (essential for Docker and AI frameworks), it introduces a distinct memory management challenge: the VM has its own memory manager, independent of the Windows memory manager.

By default, WSL2 is configured to consume up to 50% of the host's total RAM (i.e., 8GB on a 16GB system) or 80% on older builds.[13] While this allocation seems sufficient, the dynamic allocation mechanism often fails to release memory back to Windows aggressively enough. Linux, by design, will use free RAM to cache disk I/O (the Page Cache). To the Windows host, this cached memory appears as "in use" by the Vmmem process, leading to a situation where WSL2 holds 8GB (mostly cache), Windows holds 6GB, and the user is left with no headroom for the browser or IDE.[15]

3.1. The .wslconfig Directive: Hypervisor Tuning

The behavior of the WSL2 VM is governed by the .wslconfig file located in the user's home directory (%UserProfile%). For a 16GB system targeting AI workloads, the default dynamic allocation is suboptimal. Setting a hard limit forces the Linux kernel to manage its own memory pressure more actively, invoking its own OOM killer or reclaiming cache before requesting more from the host.

Recommended Configuration for 16GB Systems:

Ini, TOML

```
[wsl2]
# Constraint: Limit RAM to 6GB.
# Reasoning: Reserves ~10GB for Windows (OS + VS Code + Chrome + GPU Drivers).
memory=6GB

# Swap Configuration: 4GB on NVMe.
# Reasoning: Provides overflow for loading models before quantization.
swap=4GB
swapFile=C:\\Temp\\wsl-swap.vhdx

# Processor Limit: 8 Logical Cores.
# Reasoning: Prevents WSL2 from starving the Windows UI thread during heavy compilation.
processors=8

# EXPERIMENTAL SETTINGS (Critical for 2024/2025 Builds)
[experimental]
# autoMemoryReclaim: gradual
# Reasoning: Uses cgroup v2 to detect idle pages and return them to Windows.
autoMemoryReclaim=gradual

# sparseVhd: true
# Reasoning: Automatically shrinks the virtual disk image when files are deleted.
sparseVhd=true

# networkingMode: mirrored
# Reasoning: improves compatibility and potentially reduces overhead of NAT translation.
networkingMode=mirrored
```

3.2. Advanced Memory Reclamation: autoMemoryReclaim

Introduced in late 2023 and refined in 2024/2025 updates, autoMemoryReclaim is the most significant feature for constrained WSL2 environments. It utilizes Linux Control Groups (specifically cgroup v2) to detect memory pages that are allocated but idle (such as cold file caches) and proactively returns them to the Windows host.[15]

- **Gradual vs. Dropcache:** The gradual setting is preferred over dropcache. dropcache triggers a periodic echo 1 > /proc/sys/vm/drop_caches, which can cause severe performance jitter by evicting hot data that is immediately needed again. gradual offers a smoother reclamation curve that balances host availability with guest performance.[16]
- **Stability Concerns:** Some users have reported that autoMemoryReclaim can cause Docker Desktop to freeze if the container engine enters a "Resource Saver" mode while memory is being reclaimed.[18] If stability issues arise (e.g., WSL hanging), reverting to a fixed memory limit without auto-reclaim is the rollback strategy.

3.3. Swap Strategy on NVMe

Given the 16GB hard limit, swap space acts as the critical overflow buffer. On the ThinkPad X1 Extreme Gen 4i, the NVMe drive (likely PCIe Gen 4) offers read/write speeds that make swapping significantly less penalizing than on older SATA SSDs.

- **Swap Sizing:** A 4GB swap file in WSL2 provides a total virtual memory space of 10GB (6GB RAM + 4GB Swap). This is sufficient to load larger model weights (e.g., a 7GB FP16 model) momentarily before quantization or offloading, preventing immediate process termination.[19]
- **Page File (Windows):** For the Windows host, the "System Managed" size is generally best, but setting a fixed minimum (e.g., 4GB) ensures that Windows can offload its own idle pages to disk to make room for the active WSL2 workload.[21]

3.4. The "8 Second Rule" and Persistence

Changes to .wslconfig do not apply instantly. The WSL2 VM must be completely terminated. The "8 Second Rule" dictates that after running wsl --shutdown, one must wait approximately 8 seconds for the underlying Hyper-V container networking and memory subsystems to fully halt before restarting. Failure to wait results in the old configuration persisting, which can lead to confusion during testing.[19]

---

4. Application Layer Optimization: IDEs and Browsers

With the OS and Hypervisor tuned, the next layer of inefficiency lies in the user-space applications: the IDE (VS Code) and the Web Browser. Both typically utilize the Electron framework or Chromium engine, which are notoriously memory-hungry due to their multi-process architecture.

4.1. Visual Studio Code: Taming the Electron Giant

Visual Studio Code (VS Code) spawns multiple helper processes (Code Helper (Renderer) and Code Helper (Plugin)). In an AI development context, the most significant resource drain is often the file watcher service and the Python language server (Pylance).

4.1.1. File Watcher Optimization

VS Code uses a file watcher (often based on chokidar) to monitor the workspace for changes. In AI projects, datasets can contain thousands or millions of small image or text files. If VS Code attempts to "watch" these for changes, memory usage spikes, and the CPU consumes power indexing them.

Configuration Strategy (settings.json):
Users must aggressively exclude large directories from the watcher using files.watcherExclude. This prevents the "Renderer" process from consuming 1-2GB of RAM during project indexing.5

JSON

```json
"files.watcherExclude": {
  "**/.git/objects/**": true,
  "**/.git/subtree-cache/**": true,
  "**/node_modules/**": true,
  "**/venv/**": true,
  "**/__pycache__/**": true,
  "**/dataset/**": true,  // Critical for AI/ML datasets
  "**/data/**": true      // Generic data folder exclusion
}
```

4.1.2. Extension Hygiene and Workspace Isolation

Extensions in VS Code run in a separate host process. A Python extension, Pylance, Jupyter, and GitLens running simultaneously can easily consume 2GB.

- **Workspace-Specific Enabling:** Do not enable all extensions globally. Use the "Disable (Workspace)" or "Enable (Workspace)" feature. For a pure Python/AI workspace, disable unrelated extensions like Docker (unless active), Kubernetes, or frontend tools (ESLint/Prettier).[24]
- **Pylance Memory:** In settings.json, ensure python.analysis.memory.keepLibraryAst is set to false (if available) or limit the indexing depth to prevent Pylance from caching the Abstract Syntax Tree of every library in the environment.

4.2. Browser Optimization: Memory Saver vs. Sleeping Tabs

Modern browsers (Chrome/Edge) offer mechanisms to freeze inactive tabs, which is essential when researching with 20+ tabs open.

- **Chrome Memory Saver:** This feature frees up memory from inactive tabs. The "Balanced" or "Maximum" setting ensures that tabs not touched in 5-10 minutes are evicted from RAM.[26]
- **Edge Sleeping Tabs:** Edge's implementation is often cited as superior to Chrome's due to deeper OS integration on Windows. It can reduce memory usage of sleeping tabs by up to 85%.[28] For a memory-constrained Windows 11 system, Edge is technically the more efficient browser choice, though Chrome with Memory Saver enabled is a viable alternative.

4.3. Startup and Background Process Audit

Tools like **Sysinternals Autoruns** are indispensable for identifying hidden startup processes that do not appear in the standard Task Manager "Startup" tab.

- **Action:** Run Autoruns as Administrator. Check the "Logon", "Scheduled Tasks", and "Services" tabs. Uncheck entries related to update assistants (Adobe, Java, Google), telemetry agents, or unused peripheral drivers.[29]
- **Hidden Processes:** Look for "Updater" services (e.g., LGHUBUpdaterService, AdobeARMservice). These often run constantly to check for updates. Disabling them and updating manually is a significant RAM saver.[4]

---

5. AI/ML Workload Adaptation: The 4GB VRAM Limit

The NVIDIA RTX 3050 Ti with 4GB VRAM is the hard constraint for this system. Attempting to load standard models (e.g., Whisper Large V2 in FP16 requires ~3GB, leaving little room for overhead) will result in frequent OOM errors or spillover into system RAM (Shared GPU Memory). Shared GPU memory allows the GPU to use system RAM when VRAM is full, but the PCIe bus latency (even Gen 4) is significantly slower than VRAM, causing inference speed to plummet.[31] The strategy must focus on keeping the working set *entirely* within the 4GB VRAM.

5.1. Whisper STT: Quantization and Runtime Selection

OpenAI's Whisper model comes in various sizes. The "Large" model is desired for accuracy but is unwieldy on 4GB VRAM in its standard precision (FP32 or FP16).

Comparative Analysis of Runtimes:

Implementation	Precision	VRAM Usage (Approx)	Performance Note	Recommendation

OpenAI Whisper (PyTorch)	FP16	~4.7GB - 11GB	High overhead. Likely to OOM on 3050 Ti.[32]	Avoid
Faster-Whisper (CTranslate2)	Int8	**~2.9GB**	Uses CTranslate2 engine. Fits comfortably in 4GB. Minimal accuracy loss.	**Primary Choice**
Whisper.cpp	Int8 / Q5	~1.5GB (System RAM)	Runs on CPU. Extremely efficient but slower than GPU.	Backup Choice
Distil-Whisper	FP16	~2-3GB	Smaller model (distilled). Faster, but may have lower accuracy on complex audio.[33]	Alternative

Recommendation: Use faster-whisper with int8 quantization. This library uses CTranslate2, a fast inference engine for Transformer models that supports 8-bit quantization.

Python

```python
from faster_whisper import WhisperModel
# compute_type="int8" is the critical parameter for <4GB VRAM
model = WhisperModel("large-v3", device="cuda", compute_type="int8")
```

This configuration consumes approximately 2.9GB VRAM, leaving ~1.1GB for the display buffer (if the laptop is using the dGPU for display) and other OS overhead.[32]

5.2. Kokoro TTS: ONNX Runtime Efficiency

Kokoro is a high-quality, lightweight TTS model (82M parameters). While small, running it in raw PyTorch incurs the overhead of the full PyTorch framework library loaded into memory, which can be hundreds of megabytes.

- **ONNX Runtime:** Converting Kokoro to ONNX format (or using kokoro-onnx) decouples the model from the heavy PyTorch dependency. The ONNX runtime is significantly lighter and optimizes graph execution.[34]
- **Resource Usage:** Kokoro-ONNX on a GPU uses <3GB VRAM and is faster than real-time. Crucially, on CPU, it uses only ~300MB RAM and runs faster than real-time on modern CPUs.
- **Strategic Offloading:** To save VRAM for Whisper, **run Kokoro on the CPU**. The i7/i9 in the ThinkPad X1 Extreme is powerful enough to handle TTS generation without stalling the GPU, allowing the 3050 Ti to remain dedicated to the heavier STT tasks.[34]

5.3. Embedding Models and Vector Stores

Embedding models (e.g., all-MiniLM-L6-v2 or e5-small) are generally small (<1GB). However, ensuring they do not compete for VRAM is key.

- **CPU Offloading:** Since embedding generation for single queries is fast, running these on the CPU (using sentence-transformers with device='cpu') is a strategic trade-off.
- **Quantized Embeddings:** Libraries like optimum or onnxruntime can run quantized versions of embedding models, further reducing CPU RAM usage. Using int8 quantization for embeddings can speed up computation on the CPU by 3x.[36]

---

6. GPU Driver Optimization and Hardware Tuning

The software stack relies on the underlying driver for memory management. Standard NVIDIA drivers come with telemetry and "GeForce Experience" bloat which consumes system RAM.

6.1. Driver Installation via NVCleanInstall

The standard "Express" installation of NVIDIA drivers includes components irrelevant to AI, such as the 3D Vision Controller, Shield Streaming service, and Telemetry.

- **Tool:** NVCleanInstall is recommended to strip the driver installer.
- **Configuration:** Select *only* "Display Driver", "PhysX" (needed for some games/simulations, low impact), and "CUDA" (Essential for AI).
- **Outcome:** Reduces the background process count by 3-5 services and frees roughly 100-200MB of RAM.[37]

6.2. Latency and Shared Memory Management

- **Low Latency Mode:** In the NVIDIA Control Panel, set "Low Latency Mode" to "On" or "Ultra". This reduces the render queue (pre-rendered frames). While primarily for gaming, it reduces the amount of memory allocated for frame buffering, effectively keeping VRAM usage leaner.[37]

- **Shared GPU Memory:** This is system RAM that Windows reserves for the GPU to use if VRAM fills up. On a 16GB system with a 4GB card, Windows might reserve ~8GB as "Shared GPU Memory". While this cannot be easily disabled, monitoring it via Task Manager is critical. If "Shared GPU Memory" usage spikes during inference, it indicates the model is too large for VRAM, and performance will degrade due to PCIe bus latency. The solution is *not* to increase shared memory usage, but to quantize the model further (e.g., Int8 to Int4) to fit back into VRAM.[40]

---

7. Implementation Guide and Automation

This section synthesizes the research into actionable steps and automated scripts.

7.1. Step-by-Step Optimization Checklist

Step	Action	Expected Savings/Impact
1	**NVIDIA Driver Clean Install**	Use NVCleanInstall (Driver + CUDA only).
2	**Registry NDU Fix**	Set HKLM\SYSTEM\ControlSet001\Services\Ndu\Start to 4.
3	**Service Debloat**	Disable SysMain, DiagTrack, WSearch, etc. (See Script A).
4	**Startup Audit**	Run Autoruns, disable updaters/bloat.
5	**WSL2 Config**	Apply .wslconfig (6GB Limit, autoMemoryReclaim).
6	**VS Code Tuning**	Apply files.watcherExclude settings.

7	**Browser Tuning**	Enable Memory Saver / Sleeping Tabs.
8	**AI Stack Setup**	Install faster-whisper (Int8) & kokoro-onnx.

7.2. PowerShell Automation Scripts

Script A: Safe Service Optimizer & NDU Fix

Run as Administrator. This script applies the registry fix and disables the safe-list services identified in Section 2.

PowerShell

```powershell
# Title: AI_Dev_Optimizer.ps1
# Description: Optimizes Services and Registry for 16GB RAM AI Dev Machine

Write-Host "Starting Optimization for AI Workloads..." -ForegroundColor Cyan

# 1. Fix NDU Memory Leak (Non-Paged Pool)
$nduPath = "HKLM:\SYSTEM\ControlSet001\Services\Ndu"
if (Test-Path $nduPath) {
  Set-ItemProperty -Path $nduPath -Name "Start" -Value 4
  Write-Host "NDU Service disabled (Fixes non-paged pool leak)." -ForegroundColor Green
} else {
  Write-Host "NDU Service key not found." -ForegroundColor Red
}

# 2. Service Optimization Safe-List
# These services are identified as safe to disable for a Dev environment
$servicesToDisable = @(
  "SysMain",        # Pre-fetching, wastes RAM on SSDs
  "DiagTrack",      # Telemetry
  "WSearch",        # Indexing, heavy I/O and RAM
  "MapsBroker",     # Downloaded Maps Manager
  "XblAuthManager", # Xbox Live Auth (if not gaming)
  "XblGameSave",    # Xbox Cloud Save
  "DPS",            # Diagnostic Policy Service
```

```
    "PcaSvc"          # Program Compatibility Assistant
)

foreach ($service in $servicesToDisable) {
    if (Get-Service $service -ErrorAction SilentlyContinue) {
        Stop-Service $service -Force -ErrorAction SilentlyContinue
        Set-Service $service -StartupType Disabled
        Write-Host "Service $service disabled." -ForegroundColor Green
    } else {
        Write-Host "Service $service not found or already disabled." -ForegroundColor Yellow
    }
}

Write-Host "Optimization Complete. A system reboot is required." -ForegroundColor Yellow
```

Script B: Automated Standby List Cleaner (Native API)

This script uses C# reflection to call the native Windows API, avoiding third-party EXEs. It can be scheduled via Task Scheduler to run on system idle.

PowerShell

```
# Title: Clean_Standby_Native.ps1
# Description: Clears Standby List using native ntdll.dll calls via C# Reflection

$code = @"
using System;
using System.Runtime.InteropServices;

public class MemoryCleaner {

    public static extern int EmptyWorkingSet(IntPtr hwProc);


    public static extern UInt32 NtSetSystemInformation(int InfoClass, IntPtr Info, int Length);

    public static void ClearStandby() {
        // SYSTEM_MEMORY_LIST_COMMAND = 80
        // Command 4 = PurgeStandbyList
        int sysInfoClass = 80;
        int command = 4;
        IntPtr ptr = Marshal.AllocHGlobal(sizeof(int));
        Marshal.WriteInt32(ptr, command);
        NtSetSystemInformation(sysInfoClass, ptr, sizeof(int));
        Marshal.FreeHGlobal(ptr);
    }
}
"@

Add-Type -TypeDefinition $code
[MemoryCleaner]::ClearStandby()
Write-Host "Standby List Purged."
```

7.3. Monitoring Tools and Maintenance

To maintain this optimized state, visibility is key.

- **RAMMap (Sysinternals):** Essential for visualizing the "Non-Paged Pool" and "Standby List." Use this to confirm the NDU fix is working (Non-Paged pool should remain stable, not growing indefinitely).
- **Process Explorer (Sysinternals):** Replaces Task Manager. Configure it to show "Private Bytes" (committed memory) and "Working Set" (physical RAM). This distinguishes between memory a process *needs* versus memory it is *holding*.[42]
- **nvtop (inside WSL2):** A command-line tool to monitor GPU usage from within Linux. Essential for checking VRAM usage of Python scripts in real-time.

Monthly Maintenance Schedule:

1. **Update Check:** Run wsl --update to ensure the latest kernel with autoMemoryReclaim fixes is installed.
2. **Disk Optimization:** Run wsl --manage <distro> --set-sparse true to ensure the VHDX file releases space back to Windows.
3. **Extension Audit:** Check VS Code for enabled extensions that are no longer needed for the current project context.
4. **Health Check:** Run sfc /scannow to ensure no Windows system files (which might cause service loops) are corrupted.

7.4. Benchmarks and Expectations

Before Optimization:

- **Idle RAM Available:** ~2.5GB
- **WSL2 Consumption:** ~8GB (Static/Ballooned)
- **Whisper Inference:** Fails (OOM) or runs on CPU (Slow).
- **System Feel:** Stuttering during heavy I/O; "Not responding" warnings.

After Optimization:

- **Idle RAM Available:** ~5.5GB - 6.0GB (NDU fix + Service Debloat + SysMain Off).

- **WSL2 Consumption:** Dynamic (Idle ~500MB, Load ~6GB). Returns memory to Windows within 30-60 seconds of process end.
- **Whisper Inference:** faster-whisper (Int8) uses ~2.9GB VRAM. Runs successfully on GPU.
- **System Feel:** Snappy application switching; Browser tabs sleep effectively.

8. Rollback Instructions

In the event of system instability, the following steps will revert changes:

1. **Re-enable Services:** Set SysMain, WSearch, and DPS back to Automatic via services.msc.
2. **Revert Registry:** Change HKLM\SYSTEM\ControlSet001\Services\Ndu\Start value back to 2 (Automatic).
3. **Reset WSL Configuration:** Delete or rename the .wslconfig file in %UserProfile% and restart WSL (wsl --shutdown). This reverts to the default 50% RAM allocation.
4. **Terminal Fix:** If the Terminal fails to launch, ensure TabletInputService is running.

By implementing this architectural restructuring, the Lenovo ThinkPad X1 Extreme Gen 4i transforms from a resource-constrained laptop into a finely tuned edge-AI development station, capable of running modern generative models within its hardware limits.

Works cited

1. How Do I Fix High Memory Usage Windows 11? - AOMEI Partition Assistant, accessed on January 4, 2026, https://www.diskpart.com/windows-11/high-memory-usage-windows-11-0001.html
2. Possible memory leak - task manager doesn't add up : r/Windows10 - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/Windows10/comments/3f93te/possible_memory_leak_task_manager_doesnt_add_up/
3. (NDU registry change) Increase performance immediately! AMAZING! - Page 10 - Hardware & Peripherals - Microsoft Flight Simulator Forums, accessed on January 4, 2026, https://forums.flightsimulator.com/t/ndu-registry-change-increase-performance-immediately-amazing/504551?page=10
4. Safety of disabling services in Windows 10 and 11 · GitHub, accessed on January 4, 2026, https://gist.github.com/Aldaviva/0eb62993639da319dc456cc01efa3fe5
5. Reducing VSCode Memory Consumption - DEV Community, accessed on January 4, 2026, https://dev.to/claudiodavi/reducing-vscode-memory-consumption-527k
6. accessed on January 4, 2026, https://smallbusiness.chron.com/uninstall-preinstalled-lenovo-bloatware-50782.html
7. What bloatware should I remove my new Lenovo laptop? Should I change any

settings? Update any drivers? - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/Lenovo/comments/zidr53/what_bloatware_should_i_remove_my_new_lenovo/
8. I found this file on a youtube guide from a safe channel and i don't know if it's 100% safe : r/antivirus - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/antivirus/comments/sgnp4p/i_found_this_file_on_a_youtube_guide_from_a_safe/
9. FIX FOR CONSTANT CRASHES IN GAME. WORKING SOLUTION...READ HERE!!! :: The Last of Us™ Part I General Discussions - Steam Community, accessed on January 4, 2026, https://steamcommunity.com/app/1888930/discussions/0/6197556602762433612/
10. Clear the windows 7 standby memory programmatically - Stack Overflow, accessed on January 4, 2026, https://stackoverflow.com/questions/12841845/clear-the-windows-7-standby-memory-programmatically
11. Would it be possible to completely remove standby memory ? - Microsoft Q&A, accessed on January 4, 2026, https://learn.microsoft.com/en-us/answers/questions/3922350/would-it-be-possible-to-completely-remove-standby
12. IgorMundstein/WinMemoryCleaner: This free RAM cleaner uses native Windows features to optimize memory areas. It's a compact, portable, and smart application. - GitHub, accessed on January 4, 2026, https://github.com/IgorMundstein/WinMemoryCleaner
13. How to configure memory limits in WSL2 - Willem's Fizzy Logic, accessed on January 4, 2026, https://fizzylogic.nl/2023/01/05/how-to-configure-memory-limits-in-wsl2
14. WSL 2 does not have all memory available to it - Super User, accessed on January 4, 2026, https://superuser.com/questions/1707758/wsl-2-does-not-have-all-memory-available-to-it
15. Docker Desktop Using Too Much Memory (VmmemWSL): Cause and Solution - DevOps.dev, accessed on January 4, 2026, https://blog.devops.dev/docker-desktop-using-too-much-memory-vmmemwsl-cause-and-solution-0a54998ab65a
16. Windows Subsystem for Linux September 2023 update - Microsoft Dev Blogs, accessed on January 4, 2026, https://devblogs.microsoft.com/commandline/windows-subsystem-for-linux-september-2023-update/
17. WSL is nice, but it takes up a lot of RAM on my system. I prefer to use MSYS2.... | Hacker News, accessed on January 4, 2026, https://news.ycombinator.com/item?id=39813522
18. Resource Saver mode / pausing causes WSL to freeze · Issue #14656 · docker/for-win, accessed on January 4, 2026, https://github.com/docker/for-win/issues/14656
19. Advanced settings configuration in WSL | Microsoft Learn, accessed on January

4, 2026, https://learn.microsoft.com/en-us/windows/wsl/wsl-config
20. Docker Desktop eats RAM past WSL limit set in .wslconfig file, accessed on January 4, 2026, https://forums.docker.com/t/docker-desktop-eats-ram-past-wsl-limit-set-in-wslconfig-file/128150
21. Pagefile size? | Windows 11 Forum, accessed on January 4, 2026, https://www.elevenforum.com/t/pagefile-size.38551/
22. What's the virtual memory:RAM size ratio for Windows 11? : r/Windows11 - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/Windows11/comments/1dyq368/whats_the_virtual_memoryram_size_ratio_for/
23. My VSCode Settings to Make VSCode 10x Faster - Apidog, accessed on January 4, 2026, https://apidog.com/blog/vscode-settings-to-make-vscode-10x-faster/
24. Extension Marketplace - Visual Studio Code, accessed on January 4, 2026, https://code.visualstudio.com/docs/configure/extensions/extension-marketplace
25. Is there a way to install vs-code extensions only inside the workspace? - Super User, accessed on January 4, 2026, https://superuser.com/questions/1449639/is-there-a-way-to-install-vs-code-extensions-only-inside-the-workspace
26. Personalize Chrome performance - Google Help, accessed on January 4, 2026, https://support.google.com/chrome/answer/12929150?hl=en
27. Chrome now shows each active tab's memory usage! | by Addy Osmani - Medium, accessed on January 4, 2026, https://medium.com/@addyosmani/chrome-now-shows-each-active-tabs-memory-usage-4f74876538e6
28. Is Chrome's recent tab-sleep functionality actually *worse - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/chrome/comments/12dtao9/is_chromes_recent_tabsleep_functionality_actually/
29. How to Debloat Windows 11 to get more Performance for gaming - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/OptimizedGaming/comments/1facx6u/how_to_debloat_windows_11_to_get_more_performance/
30. More FPS? NDU Registry Tweak. :: Microsoft Flight Simulator (2020) English - General Discussions - Steam Community, accessed on January 4, 2026, https://steamcommunity.com/app/1250410/discussions/0/3422186614297671551/?l=hungarian&ctp=1
31. Is there any way I can improve my 4gig vram 3050 : r/GamingLaptops - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/GamingLaptops/comments/1ize9hn/is_there_any_way_i_can_improve_my_4gig_vram_3050/
32. faster-whisper/README.md at master - GitHub, accessed on January 4, 2026, https://github.com/SYSTRAN/faster-whisper/blob/master/README.md
33. How to run Whisper Large-v3 on 4gb vram (in my case, 1050 Ti) : r/LocalLLaMA - Reddit, accessed on January 4, 2026,

https://www.reddit.com/r/LocalLLaMA/comments/1bbqpes/how_to_run_whisper_largev3_on_4gb_vram_in_my_case/

34. Introcuding kokoro-onnx TTS : r/LocalLLaMA - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/LocalLLaMA/comments/1htwkba/introcuding_kokoroonnx_tts/
35. Kokoro TTS MCP Server - LobeHub, accessed on January 4, 2026, https://lobehub.com/mcp/your-username-kokoro-tts-mcp
36. How to compute LLM embeddings 3X faster with model quantization - Medium, accessed on January 4, 2026, https://medium.com/nixiesearch/how-to-compute-llm-embeddings-3x-faster-with-model-quantization-25523d9b4ce5
37. smo0ths/Installing-and-optimizing-new-nvidia-drivers-on-windows-11-gaming-PC - GitHub, accessed on January 4, 2026, https://github.com/smo0ths/Installing-and-optimizing-new-nvidia-drivers-on-windows-11-gaming-PC
38. Get Higher FPS Lower Latency Using CUSTOM Nvidia Geforce RTX Drivers With NVCleanstall APP - YouTube, accessed on January 4, 2026, https://www.youtube.com/watch?v=CHdAOUka31Y
39. Here's my settings for getting the best results with G-Sync (Others chime in to verify or add on) : r/losslessscaling - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/losslessscaling/comments/1i7xjd8/heres_my_settings_for_getting_the_best_results/
40. Can 4GB VRAM run Stellar Blade? RTX 3050 Laptop | Optimization Guide - YouTube, accessed on January 4, 2026, https://www.youtube.com/watch?v=7NphU64fuq8
41. How Can I Make the Most of My RTX 3050 (4GB) for Machine Learning in 2025? - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/learnmachinelearning/comments/1is6ii7/how_can_i_make_the_most_of_my_rtx_3050_4gb_for/
42. Debloat Windows 11 or not?, accessed on January 4, 2026, https://www.elevenforum.com/t/debloat-windows-11-or-not.40725/