Zusammenfassung

Im Mittelpunkt stehen Green Coding, Green IT, nachhaltige Software, Open-Source-Software und OpenCode. Ziel ist es, ökologische und ökonomische Anforderungen in der Softwareentwicklung, im Betrieb von IT-Infrastruktur und in der öffentlichen Verwaltung stärker miteinander zu verbinden.

Nachhaltige digitale Infrastruktur und Software agieren zusammen Stromverbrauch, Hardwarebelastung und Materialeinsatz reduzieren. Die Nachhaltigkeit der Rechenzentren könnte durch Abwärmenutzung und Wiederverwendung von Hardware zusätzlich erhöht werden.

Zusätzlich kann die Nachhaltigkeit erhöht werden in dem der Energieverbrauch durch neue Hardware erhöht werden, eine höhere Nachhaltigkeit kann auch durch die längere Nutzung der Hardware erreicht werden, welches sowohl Ressourcenschonender & Kostenschonender ist.

Langlebige und Effiziente IT-Systeme können steigende Anforderungen, etwa durch KI-Anwendungen, wirtschaftliche und ökologisch besser bewältigen.

Ein weiterer wichtiger Aspekt ist die Digitale Souveränität, welche mithilfe von Open-Source-Software erreicht wird, die Transparenz erhöht und die Kontrolle über Daten und Prozesse von einzelnen Herstellern & Staaten entfernt.

Der Blaue Engel für Software definiert Kriterien für ressourcen- und energieeffiziente Software, etwa Energieverbrauch, Hardware-Nutzungsdauer, Transparenz und Nutzungsautonomie. Der Blaue Engel für Rechenzentren bewertet energieeffizienten und ressourcenschonenden Betrieb, darunter Energiemanagement, Monitoring, Wiederverwendung von Hardware und erneuerbare Energieversorgung. Das Green Metrics Tool ermöglicht die Messung von Energieverbrauch, CO_2-Emissionen und Software Carbon Intensity und kann in DevOps-Prozesse integriert werden. Das Projekt AwaNetz fördert die Nutzung von Abwärme aus Rechenzentren für klimafreundliche Wärmeversorgung. OpenCode und ZenDiS unterstützen die öffentliche Verwaltung bei der rechtssicheren Nutzung und Weiterentwicklung von Open-Source-Lösungen.

Die Annahme, Nachhaltigkeit bremse Innovation wird wiedersprochen. Im Gegenteil: Nachhaltige Software zwingt zu klareren Anforderungen, bewussteren Technologieentscheidungen und effizienterem Code. Auch wenn Anfangsinvestitionen höher sein können, entstehen langfristig Vorteile durch geringeren Ressourcenverbrauch, bessere Wartbarkeit und niedrigere Betriebskosten. Performance und Energieeffizienz schließen sich nicht aus, sondern können durch gute Architektur und Optimierung gemeinsam verbessert werden.

Green Coding beschreibt den Ansatz, Software so zu entwickeln, dass sie möglichst wenig Energie verbraucht und vorhandene technische Ressourcen effizient nutzt. Während „grüne IT“ häufig den nachhaltigen Betrieb von Hardware, Rechenzentren und Cloud-Infrastrukturen meint, konzentriert sich Green Coding besonders auf die Software selbst. Ziel ist es, durch bewusste Architekturentscheidungen, effiziente Programmierung und passende Betriebsplattformen den Energieverbrauch digitaler Anwendungen zu senken. Dadurch kann nicht nur der CO_2-Ausstoß reduziert werden, sondern auch die vorhandene Hardware länger genutzt werden, weil weniger Rechenleistung, Speicher und Netzwerkressourcen benötigt werden.

Ein wichtiger Ausgangspunkt ist die Unternehmensarchitektur. Sie sollte Nachhaltigkeit nicht nur als freiwilliges Zusatzthema betrachten, sondern Green Coding als unternehmensweite nichtfunktionale Anforderung festlegen. Das bedeutet, dass Energieeffizienz ähnlich wie Sicherheit, Wartbarkeit oder Performance bereits bei der Planung eines Systems berücksichtigt werden muss. Besonders relevant ist dies bei großen Anwendungen, Cloud-Systemen oder Software mit vielen parallelen Nutzern, da dort Einsparungen deutlich stärker wirken als bei kleinen Spezialanwendungen. Unternehmen können zusätzlich durch grüne Rechenzentren, den Einsatz von Ökostrom, die Nutzung von Abwärme und intelligente Lastplanung ihren ökologischen Fußabdruck verringern. Cloud-Lösungen können dabei Vorteile bringen, wenn sie effizient betrieben werden und erneuerbare Energien nutzen.

Auch die IT-Architektur spielt eine zentrale Rolle. Bevor optimiert wird, müssen zunächst die Komponenten identifiziert werden, die den größten Energiebedarf verursachen. Dabei geht es nicht nur um einzelne Programme, sondern auch um Datenbanken, Schnittstellen, Netzwerke, Server und externe Dienste. Besonders sinnvoll sind Architekturen, die flexibel skalieren und sich bei geringer Last herunterfahren oder reduzieren lassen. Lose gekoppelte Systeme und reaktive Programmiermodelle können helfen, Lastspitzen besser zu verteilen und Ressourcen nur dann zu verwenden, wenn sie wirklich gebraucht werden. Energieeffizienz entsteht also nicht erst beim Schreiben einzelner Codezeilen, sondern bereits durch die Frage, wie ein System grundsätzlich aufgebaut ist.

Besonders deutlich wird Green Coding in der Anforderungsanalyse. Jede Funktion, die entwickelt wird, verursacht Aufwand in der Entwicklung, im Betrieb und in der Wartung. Deshalb sollten Anforderungen kritisch geprüft werden: Muss eine Anwendung wirklich Echtzeitverarbeitung bieten? Müssen Inhalte ständig dynamisch aktualisiert werden? Wird ein Feature tatsächlich gebraucht oder entsteht dadurch nur zusätzliche Komplexität? Das effizienteste Feature ist, dass gar nicht erst implementiert werden muss. Dadurch werden unnötige Berechnungen, Datenübertragungen und spätere Wartungsaufwände vermieden. Gleichzeitig muss immer ein sinnvoller Ausgleich

zwischen den fachlichen Anforderungen des Kunden und dem Ziel eines möglichst geringen Energieverbrauchs gefunden werden.

In der konkreten Entwicklung sind effiziente Algorithmen ein zentrales Mittel zur Einsparung von Energie. Ein schlecht gewählter Algorithmus benötigt mehr CPU-Zeit, mehr Speicher und oft auch mehr Datenbank- oder Netzwerkzugriffe. Deshalb sollte immer geprüft werden, ob bereits bekannte und optimierte Algorithmen genutzt werden können, anstatt eigene Lösungen zu entwickeln. Zusätzlich können einfache Entscheidungen große Wirkung haben, Berechnungsformeln sind häufig effizienter als Schleifen, Näherungsverfahren können ausreichen, wenn keine exakte Berechnung notwendig ist, und unnötige Wiederholungen sollten vermieden werden. Performance und Energieeffizienz hängen hier eng zusammen, denn schneller und ressourcenschonender Code verbraucht in der Regel auch weniger Strom.

Ein weiterer wichtiger Punkt ist die Reduzierung von Serveranfragen und Datenvolumen. Jede Anfrage an einen Server verursacht Netzwerkverkehr, Verarbeitung auf dem Server und häufig Datenbankzugriffe. Deshalb sollten Anfragen gebündelt, unnötige Roundtrips vermieden und häufig genutzte Daten sinnvoll zwischengespeichert werden. Caching verbessert nicht nur die Performance, sondern reduziert auch den Energieverbrauch. Ebenso sollte das übertragene Datenvolumen möglichst klein gehalten werden. Schlanke Dateiformate wie JSON können gegenüber umfangreicheren Formaten wie XML Vorteile bieten. Komprimierung spart zusätzlich Bandbreite, insbesondere bei Netzwerkübertragungen. Bei Medien können geeignete Formate wie JPEG, AAC oder AV1 die Datenmenge deutlich verringern.

Auch Datenbanken sind für Green Coding relevant. Fehlende oder ungeeignete Indizes führen dazu, dass Abfragen länger dauern und mehr CPU- sowie Ein-/Ausgabe-Last verursachen. Mit wachsender Datenmenge steigt dadurch auch der Energieverbrauch. Entwickler sollten daher passende Datenbankindizes verwenden und Abfragen mithilfe von Explain Plans analysieren. Diese zeigen, wie ein Datenbanksystem eine Abfrage tatsächlich ausführt und welche Kosten dabei entstehen. So lassen sich SQL-Statements gezielt optimieren. Besonders bei großen Anwendungen kann eine effizientere Datenbankabfrage erhebliche Auswirkungen auf Performance, Serverlast und Stromverbrauch haben.

Das Prinzip des Zero-Waste-Code fordert, nur den Code auszuliefern, der tatsächlich benötigt wird. Moderne Software nutzt häufig viele externe Bibliotheken und Frameworks. Diese sind nützlich, können aber auch dazu führen, dass große Mengen ungenutzter Code mitgeliefert werden. Besonders bei Webanwendungen ist das problematisch, weil der Code bei neuen Sitzungen heruntergeladen und im Browser verarbeitet werden muss. Tools wie Tree Shaking in JavaScript-Bundlern oder Bytecode-Optimierungen in Java-Umgebungen können ungenutzten Code entfernen und

Anwendungen schlanker machen. Auch Plattformen wie GraalVM können helfen, Anwendungen effizienter zu starten und den Ressourcenverbrauch zu senken.

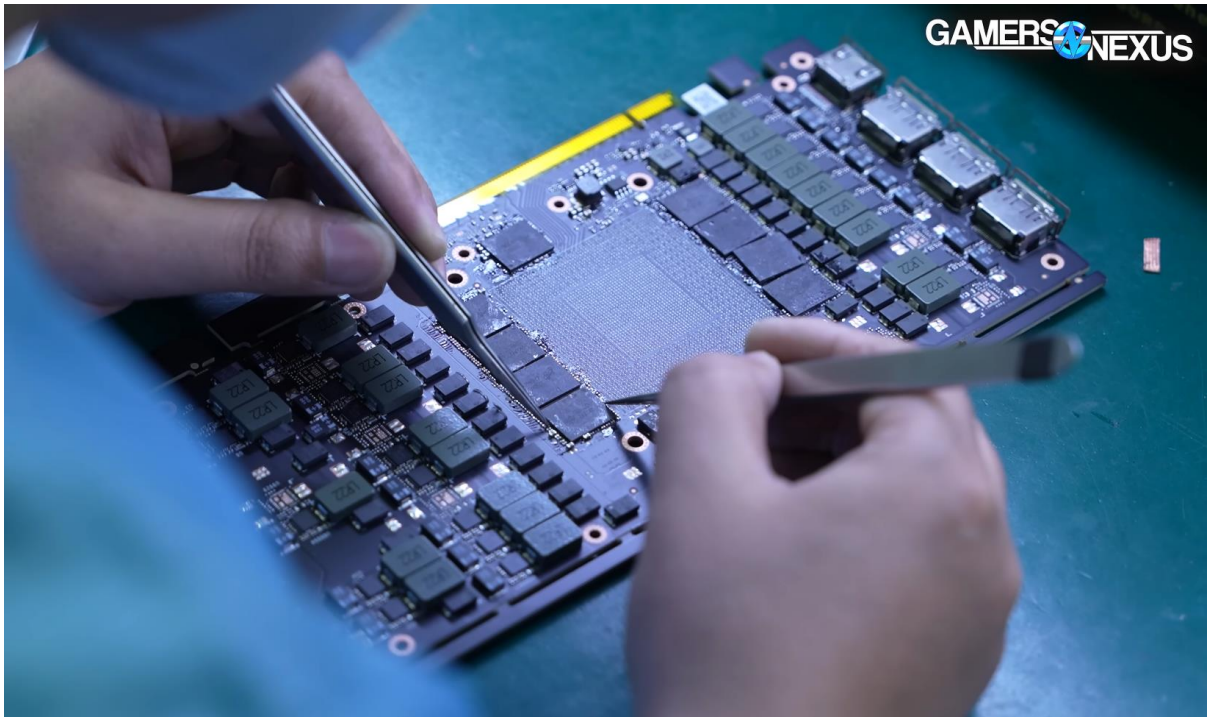
Damit Green Coding nicht nur theoretisch bleibt, muss der Energieverbrauch gemessen werden. Das Booklet stellt verschiedene Möglichkeiten vor, etwa Messungen auf Prozessebene mit Windows-Werkzeugen, Hardware-Monitoring-Tools wie Libre Hardware Monitor oder HWiNFO sowie spezialisierte Werkzeuge wie jPowerMonitor. Messungen können auf verschiedenen Ebenen erfolgen: für ein gesamtes System, einen Prozess, eine Komponente oder sogar einzelne Methoden und Tests. Wichtig ist, dass klar definiert wird, was gemessen wird, wann gemessen wird und mit welchem Werkzeug die Messung erfolgt. Nur durch nachvollziehbare Messwerte lässt sich beurteilen, ob eine Optimierung tatsächlich Energie spart.

Auch Testing, DevOps und Betrieb müssen nachhaltig gestaltet werden. Tests sind wichtig für Qualität, verursachen aber ebenfalls Energieverbrauch, besonders wenn komplette Test-Suites sehr häufig ausgeführt werden. Deshalb sollte überlegt werden, welche Tests wann notwendig sind und ob Test Impact Analysis eingesetzt werden kann, um nur die betroffenen Tests auszuführen. In DevOps-Pipelines können unnötige Builds und Testläufe vermieden werden. Im Betrieb sind containerbasierte Plattformen mit dynamischer Skalierung sinnvoll, weil sie Ressourcen an die tatsächliche Last anpassen können. Systeme sollten bei Leerlauf oder zu Randzeiten herunterfahren können, statt dauerhaft mit voller Kapazität zu laufen.

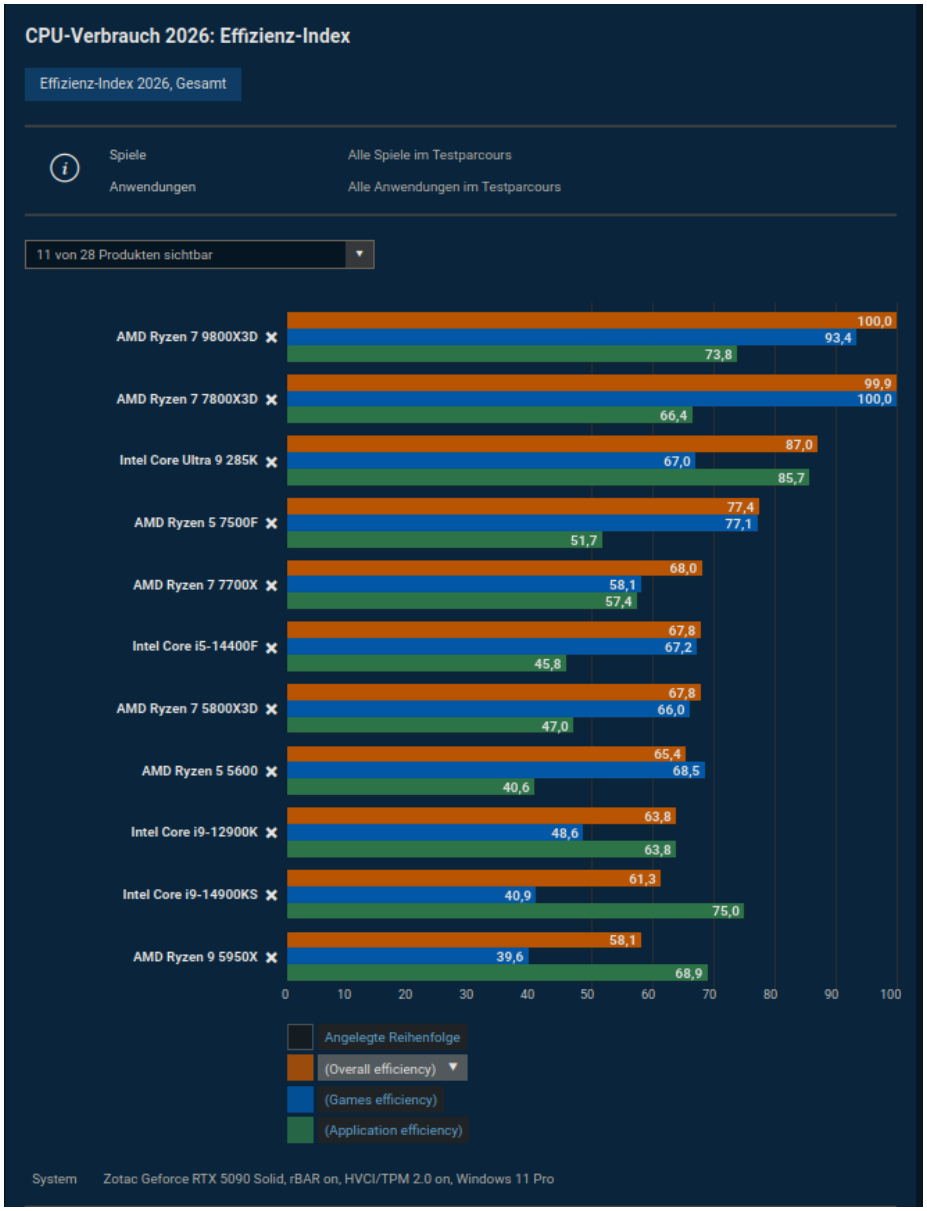
Anhang

Q1 Solaranlage

Q2 Recycle Bin Arten



Q3 Brother Zhang’s Repair Shop (张哥)


CPU-Verbrauch 2026: Effizienz-Index
Effizienz-Index 2026, Gesamt
Spiele
Alle Spiele im Testparcours
Anwendungen
Alle Anwendungen im Testparcours
11 von 28 Produkten sichtbar
AMD Ryzen 7 9800X3D
100,0
93,4
73,8
AMD Ryzen 7 7800X3D
99,9
100,0
66,4
Intel Core Ultra 9 285K
87,0
67,0
85,7
AMD Ryzen 5 7500F
77,4
77,1
51,7
AMD Ryzen 7 7700X
68,0
58,1
57,4
Intel Core i5-14400F
67,8
67,2
45,8
AMD Ryzen 7 5800X3D
67,8
66,0
47,0
AMD Ryzen 5 5600
65,4
68,5
40,6
Intel Core i9-12900K
63,8
48,6
63,8
Intel Core i9-14900KS
61,3
40,9
75,0
AMD Ryzen 9 5950X
58,1
39,6
68,9
0
10
20
30
40
50
60
70
80
90
100
Angelegte Reihenfolge
(Overall efficiency)
(Games efficiency)
(Application efficiency)
System
Zotac Geforce RTX 5090 Solid, rBAR on, HVCI/TPM 2.0 on, Windows 11 Pro

Q4 PCGH