# NAS-Box, selbstgeschnitzt

Alexander Schreiber <als@thangorodrim.ch>

http://www.thangorodrim.ch/

Chemnitzer Linux-Tage 2017, 2017-03-12

*Engineers like to solve problems. If there are no problems handily available, they will create their own problems.*

*– Scott Adams*

# Inhalt

# Über den Autor

- beschäftigt sich seit fast 20 Jahren mit Linux
- tätig als Systemingenieur bei YouTube in Zürich
- hat den einen oder anderen Computer im Haus

# Um was geht es?

- Hardware
  - Systemboards
  - Stromversorgung
  - Gehäuse
- Software
  - Storage-Service
  - weitere Dienste

# Wie alles begann...

- RAID-Array upgrade, 6x 1TB blieben übrig
- 1x $\rightarrow$ Mediaserver, restliche 5 stauben ein
- zu Schade zum Wegwerfen, Verkauf lohnt nicht
- Hmm ... in ein NAS stecken?
- fertiges 5 Bay NAS ohne HDD: ab €300 . . . 400
- ausserdem ist kaufen & Platten reinstecken langweilig

# Da kann man doch was machen...

- es gibt kleine ARM-Boards mit SATA port
- z.B. Allwinner A10/A20 basierte Boards
- aber nur ein SATA-Port
- SATA kennt das Konzept des Port-Multipliers
- vielleicht unterstützen die das...
- Plan: Hardware besorgen, testen!

# Auswahl Hauptplatine

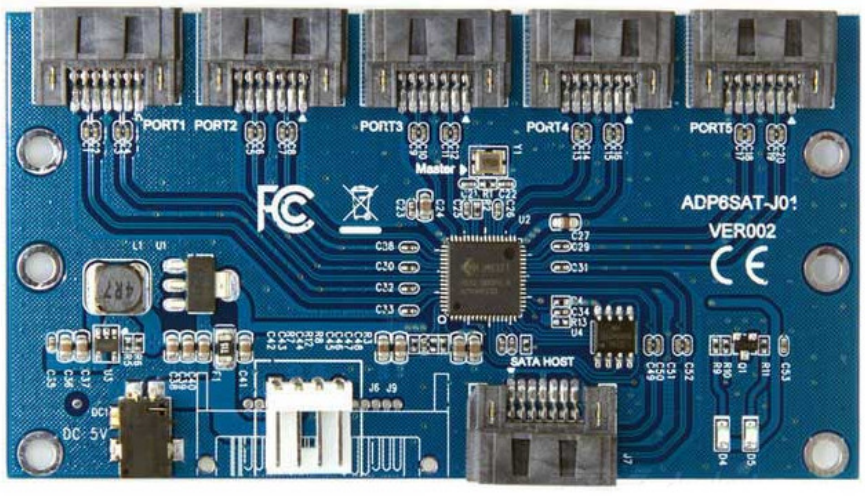
- ausgewähltes Board: Banana Pi M1
- Preis: €45 bei Amazon.de
- Allwinner A20, 2x 1 GHz Cortex A7, 1 GB RAM
- SD-Karte, SATA, echte GBit-NIC, HDMI, Strom über USB

# Exkursion: SATA-Portmultiplier

- Grundidee: 1 SATA Port → mehrere, typisch 5 . . . max. 15
- aber: mehrere Platten teilen sich 1 SATA Port Bandbreite
- Varianten:
  - Command-based switching
    - einfachster Modus
    - vergleichbar einfacher A/B Umschalter
    - nur ein Port jeweils bedient, dann weiter
    - maximale Kapazität, Leistung sub-optimal
  - FIS (Frame Information Structure) switching
    - vergleichbar USB-Hub
    - send/receive zu jedem Gerät
    - Bandbreitenverteilung
    - deutliche bessere Auslastung, aber mehr Host-CPU nötig
- → Portmultiplier: Mehr Ports, weniger Performance

# SATA-Portmultiplier, konkretes Board

- SATA-Portmultiplier: ein SATA-Port $\longrightarrow$ mehrere SATA-Ports
- benötigt Unterstützung durch SATA Host
- weit verbreitet, preiswert: JMicron JMB321
  - SATA II 3.0 GBps, eSATA
  - hot plug
  - PM/non-PM aware support
  - autonegotiation
  - ca. USD 20 .. 30 bei Amazon und DX.com



# Funktioniert es?

- laut Doku: A20 hat SATA Port-Multiplier support
- probieren geht über studieren:
  - Banana Pi M1 and Port-Multiplier
  - alle 5 Festplatten freifliegend verkabelt an Port-Multiplier
  - versorgt von externem Netzteil mit Molex-Anschluss
- $\longrightarrow$ es funktioniert, alle 5 Platten zugreifbar!
- Portmapping des Multipliers statisch, aber ... seltsam

# Soweit alles ok ...

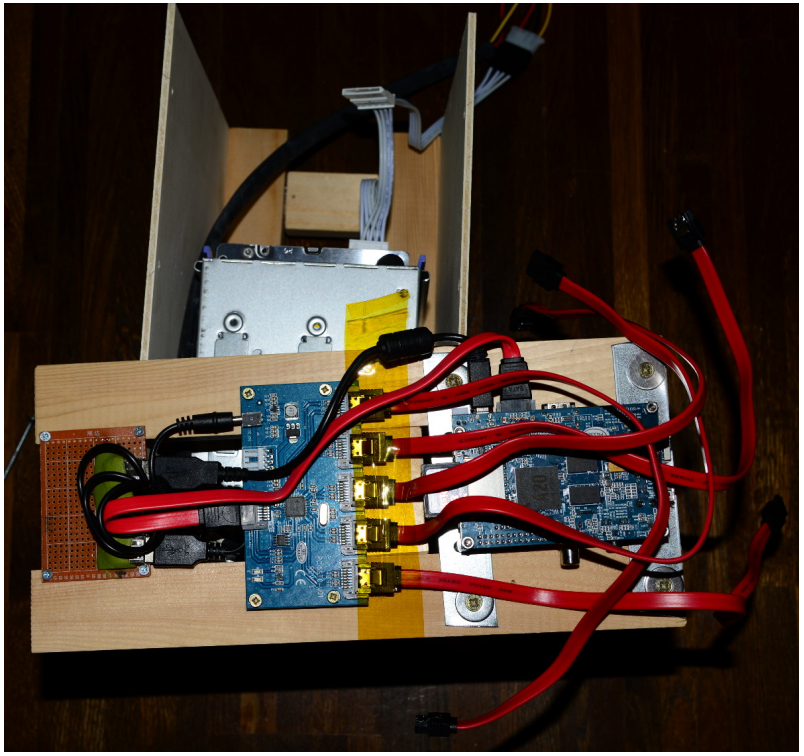
- ... oder doch nicht ganz?
- Allwinner A20
  - SATA IP Block im A20 unbalanced
  - +200 MB/s linear Lesen, 45 MB/s linear Schreiben
  - mit CPU/DRAM Takt Erhöhung etwas verbesserbar
  - Port-Multiplier support: nur Command-based switching
- JMicron JMB321 Port-Multiplier
  - nach Berichten Tendenz zu Überhitzung und dann Instabilität
  - eigene Lasttests bestätigen dies jedoch nicht
  - kann Command-based und FIS switching

# Laufwerksunterbringung

- 5 Laufwerke sind irgendwie stabil unterzubringen
- ideal: Hotplug-Käfig - aber: Preis ab EUR 100 aufwärts
- Ziel: preiswerte Datenhalde, nicht hochverfügbarer Server
- also: einfacher Laufwerkskäfig: EBay, USD 30
- dazu lautloser 120 mm Lüfter, lokaler Händler, CHF 17
- dazu 5x SATA Kabel und 1-auf-5 Molex Stromverteiler

# Das Gehäuse selbst

- Fertiggehäuse scheidet aus Prinzip aus ☺
- Also .. auf zum Baumarkt!
- Kanthölzer, Sperrholzplatten, Schrauben, Aluprofile ...
- sägen, bohren, schrauben, kleben, löten ...



# Strom muss auch noch ...1/2

- Wie das ganze mit Strom versorgen?
- gebraucht werden 12V (HDD) & 5V (HDD & Boards)
- da war doch das Netzteil mit Molex-Anschluss: 12V, 5V
- also etwas kreativ mit Molex-Verteilern basteln
- für Port-Multiplier & Banana Pi: PDU board basteln
- USB Typ A Ports, Sugru und Leerplatinen sind da $\rightarrow$ PDU
- passt, alles läuft ...

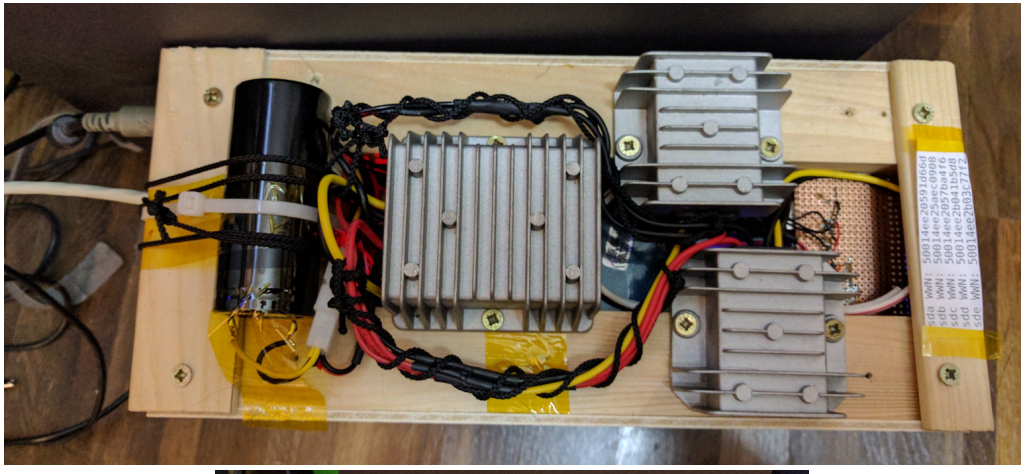
# Strom muss auch noch ...2/2

- ...bis etwa zwei Wochen später, HDDs offline (!?)
- Boards booten, aber Platten schaffen es nicht, hochzulaufen
- Netzteil überlastet, liefert weniger als 11V statt 12V
- System kommt mit 3 Platten hoch, aber ab 4 bricht die 12V Schiene auf unter 11V zusammen
- das war wohl nicht das richtige Netzteil
- aber: Linux RAID autorecovery funktioniert! (später)
- Ok, eine andere Lösung muss her

# Bestromung, zweiter Versuch

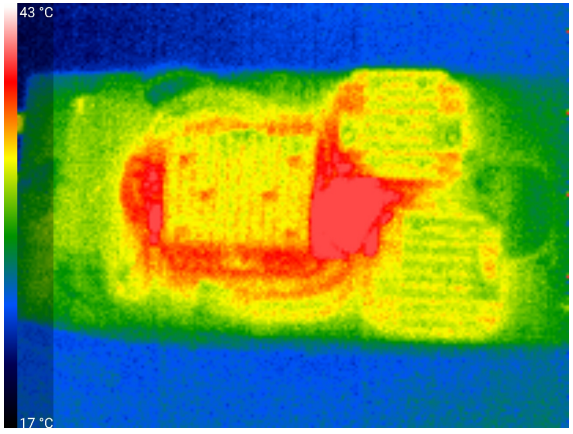
- kein passendes fertiges Netzteil gefunden
- will nicht selber mit 230V frickeln
- aber: Thinkpad Netzteil (19V) liegt rum
- Lösung:
  - Thinkpad Netzteil: 230V $\rightarrow$ 19V, lag so rum
  - 1x DC/DC Wandler (HDDs): 19V $\rightarrow$ 12V, USD 25
  - 2x DC/DC Wandler (Boards, HDDs): 19V $\rightarrow$ 5V, USD 30
  - DC/DC Wandler wollen eigentlich 20 ... 24V, geht noch
  - 12V Schiene bricht bei HDD Anlauf immer noch zusammen ...
  - 12V Schiene mit 100000 $\mu$F Kondensator gepuffert, CHF 17
- $\rightarrow$ es funktioniert, stabil!
- auch noch Monate später ☺

# Fertiges Kasterl

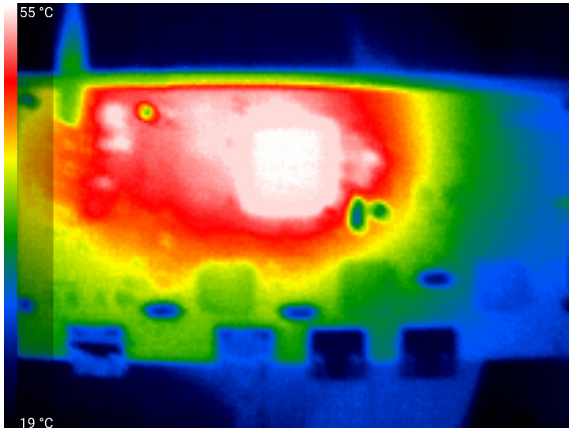



# Und das ganze im Wärmebild

DC/DC Wandler:




Banana-Pi:



# Systemsoftware

- Natürlich Linux!
- Bevorzugt Debian-basiert (weil bekannt)
- Gefunden: armbian https://www.armbian.com/
- basierend auf Debian für armhf
- Breite Unterstützung für grosse Anzahl embedded boards
- kontinuierliche Weiterentwicklung & Updates (Kernel 4.9)
- ($\mu$)SD-Karten images zum Download
- SATA-Port entweder plain SATA oder PMP-Modus
- `ahci_sunxi.enable_pmp=1` für Kernel Kommandozeile

# HDD Konfiguration

- HDDs haben fast 3 Jahre Laufzeit hinter sich
- von ursprünglich 6 starb eine nach 1 Jahr Mediaplayer Betrieb
- OS von SD-Karte läuft in Lebensdauerprobleme
- $\rightarrow$ Boot von SD-Karte, OS läuft von HDDs
- also 3 RAID-Arrays:
  - Array 1: rootfs, 8 GB, 3x RAID1 + 2x hot spare
  - Array 2: swap, 2 GB, 3x RAID1 + 2x hot spare
  - Array 3: Datenhalde, RAID6, 5 TB raw, 3 TB nutzbar
- RAID Initialisierung: Nichts für Ungeduldige ☺

# Linux automatic RAID recovery

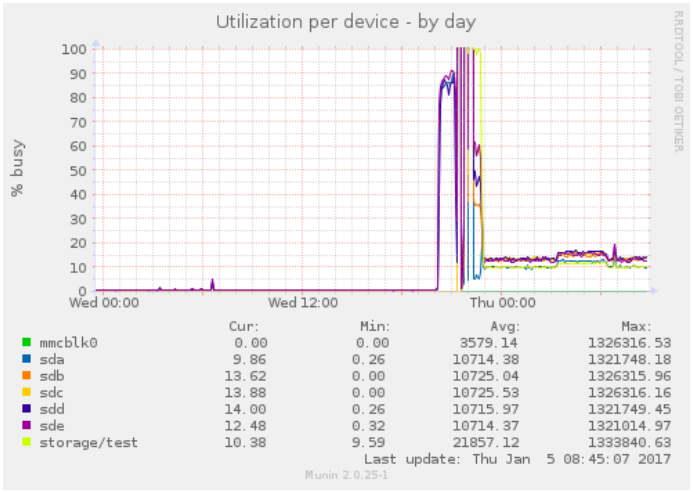
- Probleme mit Stromversorgung, nur 3 HDD online
  - rootfs & swap RAID1: 1 gültiges Element, 2 hot spares
  - RAID verwendbar, System bootet, Wiederherstellung automatisch im Hintergrund
  - Datenhalde: RAID6, gerade genug HDD online für gültiges Array
- Neubau Stromversorgung:
  - RAID6: Wiederherstellung automatisch
  - RAID1: ehemalige Daten-HDD jetzt hot spares
- selbstständige Wiederherstellung ohne Eingriff, kein Datenverlust

# Wie die Datenhalde nutzen?

- Wir haben 3 TB nutzbare Kapazität, was machen wir damit?
- NAS = **N**etwork **A**ttached **S**torage
  - zuerst: LVM2 aufsetzen, VGs, LVs → flexible Verwaltung
  - klassisch: export via Linux NFS
  - oder für Windows: Samba, Apple: Netatalk?
  - Gelegenheit zum Experimentieren:
    - iSCSI: SCSI over TCP/IP
    - AoE: ATA over Ethernet
    - sonstige: xndb, nbd
  - Nicht verfolgt: AFS, Coda: zuviel Komplexität
- Verschlüsselung?

# Überraschung vom Kernel

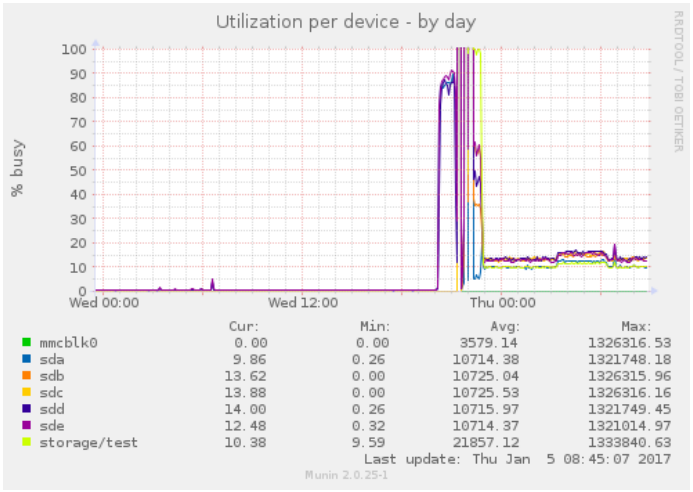
- 1 TB ext4 Dateisystem angelegt, Benchmarks gefahren
- Standardsetup: Munin, Graph sieht seltsam aus:




- Grund: mit grossen Dateisystemen macht ext4 lazy init!
- lazy init: mkfs.ext4 schreibt Basistrukturen, (grosse) inode-Tabellen nullen im Hintergrund nach erstem mount

# Überraschung vom Kernel

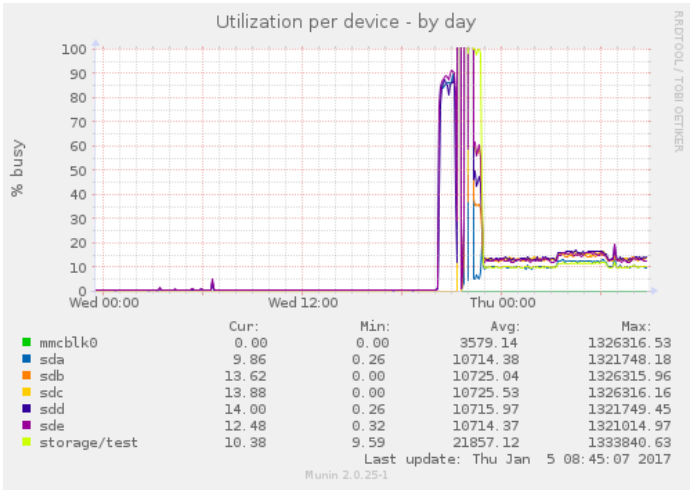
- 1 TB ext4 Dateisystem angelegt, Benchmarks gefahren
- Standardsetup: Munin, Graph sieht seltsam aus:




- Grund: mit grossen Dateisystemen macht ext4 lazy init!
- lazy init: mkfs.ext4 schreibt Basistrukturen, (grosse) inode-Tabellen nullen im Hintergrund nach erstem mount

# Überraschung vom Kernel

- 1 TB ext4 Dateisystem angelegt, Benchmarks gefahren
- Standardsetup: Munin, Graph sieht seltsam aus:




- Grund: mit grossen Dateisystemen macht ext4 lazy init!
- lazy init: mkfs.ext4 schreibt Basistrukturen, (grosse) inode-Tabellen nullen im Hintergrund nach erstem mount

# Durchsatz, lokal

- Lesen, Blockgrösse 1MB:
  - direkt `/dev/sda`: 128 MB/s
  - direkt `/dev/md2` (RAID6, 5 disks): 75 MB/s
  - verschlüsseltes Blockdevice of RAID6: 21 MB/s
- Schreiben, Blockgrösse 1MB:
  - ext4 auf RAID6: 17 MB/s
  - ext4 auf verschlüsseltem Blockdevice auf RAID6: 10 MB/s
- $\rightarrow$ asymmetrisches Verhalten SATA IP Block
- $\rightarrow$ ausbremsende Wirkung PMP + command based switching

# NFS

- Unix-Standard seit ... ewig
- v3 vs v4: single root, async, ...
- rpc.nfsd, rpc.mountd: `--no-nfs-version 4`
- Schreiben, via NFS auf ext4 auf RAID6: 15.6 MB/s
- Lesen, via NFS von ext4 auf RAID6: 42 MB/s

# Exkursion: Remote Block Protocols

- AoE, iSCSI, ...
- exportieren nicht Dateisysteme sondern direkt Blockdevices
- Standardbegriffe (von SCSI abgeleitet):
  - Initiator: sendet Kommandos, liest/schreibt Daten remote, bei Netzwerkprotokoll: Client
  - Target: empfängt Kommands, stellt Speicher zur Verfügung, bei Netzwerkprotokoll: Server
- synchrones/asynchrones Schreiben
- (nicht) routebare Protokolle

# AoE: ATA over Ethernet

- nicht routebar, ATA Blockprotokoll direkt auf Ethernet
- `aoe` Kernelmodul auf Initiator, Daemon auf Target
- Target: `vblade [-d] [-s] shelf slot netif blockdev`
- Initiator: `aoe-discover; aoe-stat ; ls /dev/ether/`
- direct I/O & sync read: 2.2 MB/s, 2+k request/s (?)
- sync read: 15 MB/s, $<$ 20 request/s $\rightarrow$ write coalescing
- Schreibtests: unzuverlässig, abgebrochen
- $\rightarrow$ AoE nicht zu empfehlen

# iSCSI: Internet SCSI

- SCSI Protokoll über TCP/IP
- IQN (iSCSI Qualified Name) Format:
  - `iqn.yyyy-mm.naming-authority:unique`
  - Beispiel: `iqn.2017-03.thangorodrim.ch:dbvol`
- Target: `apt-get install tgt`
  - Konfiguration: `/etc/tgt/conf.d/*.conf`
  - Definiert jeweils targetname, backing-store, write-cache
- Initiator: `apt-get install open-iscsi`
  - `iscsiadm --mode discoverydb -type sendtargets --portal target-hostname --discover`
  - `iscsiadm --mode node --targetname target-iqn --portal target-hostname --login`
  - Blockdevice taucht als nächstes freies `/dev/sd*` auf
  - Verbindung mit `iscsiadm ...--logout` trennen

# iSCSI, Fortsetzung

- Durchsatz:
  - Blockgrösse 1MB
  - Schreiben ext4 auf iSCSI auf RAID6: 14.6 MB/s
  - Lesen von ext4 auf iSCSI auf RAID6: 33.0 MB/s
  - Schreiben ext4 auf iSCSI auf crypted RAID6: 10.0 MB/s
  - Lesen von ext4 auf iSCSI auf RAID6: 15.9 MB/s
- iSCSI Initiator Software verbindet bei Reboot automatisch
- iSCSI hat breite Plattformunterstützung, ausgereift

# Remote Block services, Zusammenfassung

- AoE: getested, nicht zu empfehlen
- iSCSI: getestet, robust, zu empfehlen, multi-platform
- weitere Kandidaten:
  - Linux NDB: Linux only
  - Ceph RBD: eigentlich distributed ...

# Verschlüsselung

- A20 ist nicht gerade ein CPU-Riese
- dmcrypt kostet $\sim$ 50% des Durchsatzes auf A20
- Lösung:
  - Storage via iSCSI als Blockdevice exportieren
  - Verschlüsselung auf Initiator laufen lassen
  - $\rightarrow$ voller Durchsatz, minimale CPU-Last auf NAS

# Sonstige Anwendungen

- Backuphost:
  - `backup2l` und NFS/iSCSI shares
  - rsync-Server
  - git host
- low power Mediaserver
  - Video/MP3 Dateien via NFS/Samba exportieren
  - `forked-daapd`: Airplay für iDevices
  - diverse DLNA-Server: für alles andere (Android, Mediaplayer, Smart TV, ...)
  - aber: nicht viel Rechenleistung für Transcoding

# Materialliste

- Baumarkt:
  - Kantholz, Sperrholz, Schrauben, Aluprofil
  - $\sim$ CHF 15
- Elektronik
  - JMB321 basierter SATA Port Multiplier: € 21 (EBay)
  - Banana Pi M1: € 45 (Amazon)
  - DC/DC Wandler auf 12V: $ 25 (Amazon)
  - 2x DC/DC Wandler auf 5V: $ 30 (Amazon)
  - 5x 3.5" Laufwerkskäfig: $ 30 (Ebay)
  - leiser 120mm Lüfter: CHF 17 (digitec.ch)
  - Kondensator 16V 100000 $\mu$F: CHF 16.50 (conrad.ch)
  - Ladebuchse Thinkpad 19V: € 8 (EBay)
  - Thinkpad 65W Netzteil: € 8 (EBay)
  - diverse Kabelstücke, Schrumpfschlauch, Schrauben: eh da
- $\Rightarrow \sim$ € 200

# Fragen?

# Vielen Dank für Euer Interesse!